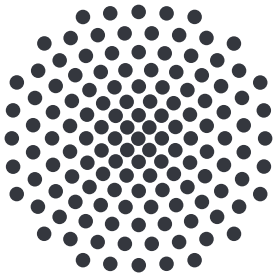
Multi-fidelity Bayesian machine learning for global optimization

Master's thesis of
Manuel Kuchelmeister

July 20, 2022

Examiner:	Prof. Dr. Jörg Main
Co-Examiner:	Prof. Dr. Christian Holm
Advisors:	Prof. Dr. Milica Todorović,
	Dr. Joakim Löfgren,
	Prof. Dr. Patrick Rinke



Aalto University
School of Science

Institute for Theoretical Physics I
University of Stuttgart
Pfaffenwaldring 57, 70550 Stuttgart
Germany

Department of Applied Physics
Aalto University
Otakaari 1 B, 02150 Espoo
Finland

Contents

1 Introduction

"Optimization is an innate human behavior" is how Roman Garnett began his book [1] on the topic of *Bayesian* Optimization (BO). It is true that optimization is a central human activity. We strive to distribute limited resources on an individual and on a collective level in such a way that we can facilitate the best possible progress. Optimization makes a fundamental contribution to social and economic progress by accomplishing this task.

At an abstract level, optimization can be defined as a decision-making process that chooses the best option from a range of options, taking a certain objective into account. And certainly many such objectives can be formulated, which can be met with very advanced optimization techniques: Researchers from Google Brain applied BO for perfecting a chocolate chip cookies recipe [2], while scientists at Meta applied multi-task BO to improve the recommender system of their social network [3].

Many other optimization problems emerge in disciplines such as natural sciences, engineering, or economics. Consequently, there is an enormous interest in providing better solutions to these problems and improving the underlying methods and algorithms. This research work deals with the development and application of a multi-fidelity artificial intelligence approach to optimize materials design and to accelerate the discovery of new structures in a quicker and more cost-efficient way.

1.1 Global optimization structure search

This work addresses the usage of Multi-fidelity Bayesian Optimization (MFBO) to accelerate the study of materials structures. Materials science deals with the study of discovering and designing new materials. One important component of this field is the study of arrangements of atoms that a material consists of. The atomic and electronic structure is determining many characteristic functional properties, and to discover and understand the structure, there are many experimental methods such as spectroscopy, diffraction with X-ray, electrons or neutrons or chemical analysis. What all these laboratory methods have in common is that they take a lot of time and can be very expensive to conduct. As there is an enormous amount of possible patterns that

can appear in a molecule, pure laboratory research for the discovery of new materials becomes rather unfeasible.

Alternatively, we can complement laboratory experiments by simulating materials. Quantum mechanics theory enables us to simulate atomistic configurations to determine accurate internal energies and properties at the microscopic and macroscopic level. The state of a materials' system is described using a set of atomic coordinates. The number of states grows vast with the number of degrees of freedom, spanning the configuration space of a system. By taking all the possible configurations for an atomistic system into account and mapping them to their energy, we obtain the so-called Potential Energy Surface (PES). The atomic structures of our interest need to be stable, and these structures reside in low-energy regions on the PES. Thus, structure search is about identifying low energy regions on the PES.

Energy calculations at the quantum mechanical level can be very time-consuming. In many cases, we are not interested in knowing the complete PES, but rather in finding and studying the stable structures that correspond to low-energy states on the PES. BO is an established probabilistic machine learning technique that is suitable for the task of finding the global minimum of a black-box function in a sample-efficient way. The phrase black-box function indicates a function that is unknown, in our case this corresponds to the 'true' PES. By the way it is constructed, the BO algorithm finds and learns low-energy regions with the objective of inferring the global minimum and is therefore well suited for the task of finding stable structures.

One essential part of BO is the construction of a regression model, often by using a distance-based measure. In systems with many degrees of freedom, it can be challenging to apply BO successfully, as there would be many samples required for a reasonable model fit. This is often not possible due to restrictions in the computational budget for the optimization. A similar problem is when the black-box function itself is very expensive to evaluate, e.g. for larger molecules. This leads to a limited number of samples being available within a given computational budget, which can complicate the successful application of BO. In order to deal with such problems, there are different optimization approaches, but defining a general-purpose universal optimization strategy is impossible [4].

This research work approaches the problem of a very expensive black-box function within BO by using a multi-fidelity optimization strategy. Multi-fidelity strategies make use of data from simulators with different accuracy levels and different evaluation costs. This allows the incorporation of information from less expensive and less accurate calculations to obtain accurate results with fewer more expensive data points. Before I introduce this topic in detail (Chapter 2), I will briefly discuss the importance that machine learning techniques have gained in materials science today.

1.2 Multi-fidelity machine learning

Machine learning is a field of computer science that uses data and algorithms to solve a predefined task. The methods developed in this field are typically used to solve prediction and decision-making tasks without having to explicitly program the underlying decision mechanisms. Instead, the mechanisms are learned by using a training algorithm, which has established machine learning as a very useful technique for the construction of models for large amounts of data.

When applied in materials science, machine learning can accelerate the discovery and design of new materials by using large available datasets, or alternatively, by guiding a growing data collection in an active learning manner [5, 6]. Active learning plays an essential role in computational materials research, as it provides a strategy for the construction of compact datasets. Direct applications of such active learning strategies inform of BO have shown to be an effective solution for tasks in materials science such as the efficient scanning of surface adsorbates [7, 8], the inference of atomistic structures in functional materials [9], the discovery of materials with multiple optimal properties by determining points on a Pareto front [10], the detection of low-energy molecular conformers of molecules [11], or for the accurate prediction of band gaps [12], to name a few.

Multi-task learning deals with the problem of predicting different tasks simultaneously, where it is assumed that the tasks are somehow related. Multi-fidelity learning is a variant of multi-task learning where the tasks describe the same property but with different levels of theory (chemical accuracy) and evaluation costs. The objective is to predict a property at the highest fidelity level available, while incorporating data of lower fidelity. Material models in computational material science are often high-dimensional, which can result in very expensive or even unfeasible calculations for larger systems, if properties such as the energies are to be determined with high accuracy. Multi-fidelity learning has shown to that by incorporating information from low accuracy data, the calculations can be significantly accelerated and thus more computationally intensive problems can be solved [10, 12–23].

Setting up such a multi-fidelity system provides us a new challenge, as there are several ways to incorporate multiple fidelities in BO [24]. In this work, I have researched a transfer learning and a multi-task learning approach and compared them against single-fidelity BO when applied on an alanine conformer search task. Conformers are molecules that consist of the same types of atoms but have different arrangements and consequently a different chemical structure. Understanding the conformation of small molecules is a problem studied in cheminformatics and computational drug discovery. There exist many methods to sample the low energy conformational space [11, 25], which aim to

find the several stable conformer structures that are associated with different electrical and chemical properties. Taking all types of degrees of freedom - bond lengths, angles and torsion - into account, would make the configuration space of conformers infeasible. As bond lengths and angles within molecules are very rigid, and can therefore can be seen as fixed, most search methods rely on finding conformers by sampling configurations with different torsion angles, so-called dihedrals [25].

The alanine system is simulated with different atomistic simulators, corresponding to different level of accuracies. This allows not only to compare the different multi-fidelity approaches, but also to test how many savings the approaches allow in solving a realistic computational materials' science problem.

1.3 Structure of this thesis

In chapter 2 I explain the mathematics behind BO, where I briefly introduce the regression model and acquisition rules that I used for the sampling strategy. I then extend the BO framework in section 2.2 to MFBO, where I explain the multi-fidelity model as well as multi-fidelity acquisition rules. This leads us to the concepts of transfer and multi-task learning in section 2.2.2 and section 2.2.3. After presenting the simulators used in this thesis in section 2.3, I introduce the Bayesian Optimization Structure Search (BOSS) library in section 2.4, which I used to implement and test my approaches.

Section 3.1 presents the overall architecture of BOSS, section 3.2 and section 3.3 highlight the for the thesis relevant and added parts of the code library, which is complemented by an explanation on how the implemented code is validated in section 3.4. The experimental design of the alanine system and an explanation of how I measured the computational savings for my calculations are found in section 3.5 and section 3.6.

The fidelity levels are compared in section 4.1. Results for single-fidelity BO baseline experiments for all three investigated levels are listed in section 4.2. These results are used to benchmark the transfer learning experiments in section 4.3 and the multi-task learning experiments in section 4.4. The approaches are discussed and compared in section 4.5. Chapter 5 rounds up this thesis by summarizing the results, a recommendation for the use of multi-fidelity approaches, and an outlook for possible further work.

2 Theory

In the previous chapter, I motivated the multi-fidelity optimization framework and its use-case for global structure search. This chapter is going to explain the theory and the relevant mathematics behind this approach, and is therefore crucial to understand the results of this thesis. In the beginning, I am going to describe the basics of BO [1]. The BO algorithm uses Gaussian Process Regression (GPR) and Acquisition Functions (AF), both will be briefly illustrated and discussed. The concepts are then extended to MFBO. In the multi-fidelity approach, several strategies to combine fidelities exist, and I will consider the ones that are implemented and tested in this thesis. To understand the computational implementation work in the following chapter, I am also going to introduce BOSS [9], a general-purpose Python package for BO, in which I implemented the multi-fidelity framework.

2.1 Bayesian optimization

Optimization tasks are formulated by using three key ingredients: an objective function, a set of variables to optimize for, and given constraints. In practice, it might not be possible to formulate the objective function with a mathematical expression, we call such optimization targets *black-box functions*. Such functions typically describe systems where we lack knowledge of underlying mechanisms or where the internal structure does not need to be considered for a specific purpose.

Black-box functions arise for instance in the design of aerospace systems [23]. Different phases in the engineering process are characterized by different levels of analysis. In the designing process, an engineer typically has to simulate or measure physical phenomena, in aerospace engineering for example related to aero- or thermodynamics. Each simulation or measurement can be done for a given design structure, where the goal is to choose a design that optimizes some objective. This objective could be related for example to the minimization of friction, resistance or the maximization of the durability of a structural element. Each experiment for a given design could potentially lead to a long measurement procedure.

In some optimization tasks, setting up and measuring the outcome of an experiment could take several days. Consequently, we desire to perform as few experiments as possible to find the optimum. Often the measurements can only be evaluated up to a (possibly unknown) level of noise, and in experimental data, we often do not have access to gradient information, preventing us from using efficient gradient-based optimizers. What simplifies many optimization procedures considerably is when the function to be optimized is convex. For a convex function, a local minimum is automatically also a global minimum. Nevertheless, no convexity property can be assumed for any black-box function. Without the condition of convexity, we have a global optimization procedure instead of a local one, for which efficient optimization routines would exist.

To overcome the many challenges associated with black-box optimization, the BO was developed. To understand the meaning of the keyword "Bayesian" in BO, I introduce the Bayesian theorem [26], which is formulated for two random variables A and B as

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}\,. \tag{2.1}$$

Here, $p(A)$ is the probability distribution of a random event A happening and $p(A|B)$ is the conditional probability distribution of the event A happening, given that event B has occurred. $p(A)$ is the so-called *prior distribution*, the original belief that an event A occurs and $p(A|B)$ is called *posterior distribution*, the (updated) belief that A occurs after incorporating information that B has been observed. For a fixed B and a variable A, the term $p(B|A)$ is called *likelihood*. This is a function that is often used to estimate unknown parameters of a probability distribution.

In the context of machine learning, the theorem can be used to update a regression model in the evidence of new samples. Machine learning is defined as the use of data to train algorithms to make classifications or predictions. In supervised machine learning, labelled samples are used to train a regression model, which is supposed to make predictions about the labels of unseen data points. *Probabilistic* machine learning methods are based on constructing a probabilistic regression model (Figure 2.1). Such a model corresponds to probability distributions trained on samples which, in addition to a prediction function (mean function $\mu(\mathbf{x})$), also learns an uncertainty function (variance function $\sigma(\mathbf{x})$) that describes the confidence of a prediction. This can be crucial in tasks like medical treatment or autonomous driving, where the information about the correctness of a prediction is as interesting as the prediction itself. Because probabilistic machine learning introduces probability distributions for modelling, the Bayesian theorem can be applied. This is what BO is based on, which is a sample-efficient strategy to optimize black-box functions and nowadays, a well established branch of probabilistic machine learning. In BO, given samples are used to construct a probabilistic model with the aim of finding the global minimum of the black-box function from which the samples were obtained.

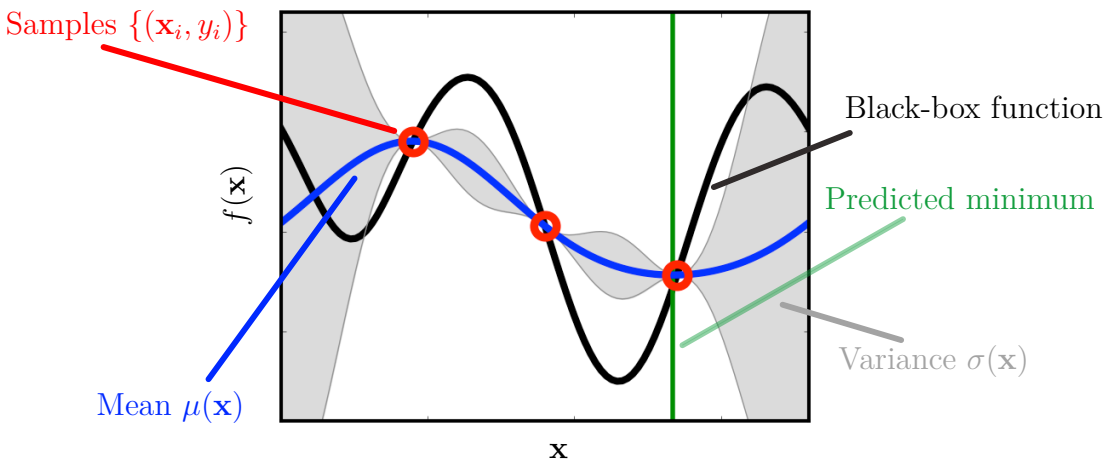


Figure 2.1: Sketch of a one dimensional probabilistic machine learning model. The variance, shown as a gray shaded area, tells us about regions in the search space where the model has a low confidence in its predictions. The mean, corresponding to the blue curve, can be used for predictions and for the inference of the minimum.

BO manifests the philosophy behind the Bayesian theorem, as the probabilistic model is updated under the observation of new samples. In its application, we start with a prior belief or distribution for the parameters of our regression model. This prior expresses our beliefs about the black-box function, before we take any samples from the function into account. The prior belief is improved using the Bayesian theorem after samples of the black-box function are observed, resulting in the posterior model. This encompasses the Bayesian approach of updating a hypothesis based on new evidence.

In the BO algorithm, new samples from the black-box function are evaluated sequentially. The policy (rule) for sampling the configuration or search space, the space spanned by the input variables of the objective function over which we want to optimize the black-box function, is encoded in the so-called AF. Minimizing the AF provides us the next sample location. The AF makes this decision based on the current state of the probabilistic model, which also takes the uncertainty into account. This is supposed to direct the search also in unknown search space regions, to make informative decisions and achieve sample efficiency. Both, the probabilistic model and the AF, will be discussed in section 2.1.1 and section 2.1.2. The typical workflow of BO is described in figure 2.2. BO starts with some initial dataset, often picked randomly or according to another sampling strategy. This data set is used to construct a regression model, which is used by the AF. Minimizing the AF determines our next sample candidate x. We evaluate the black-box function at that location and update our initial dataset with the new sample. This is

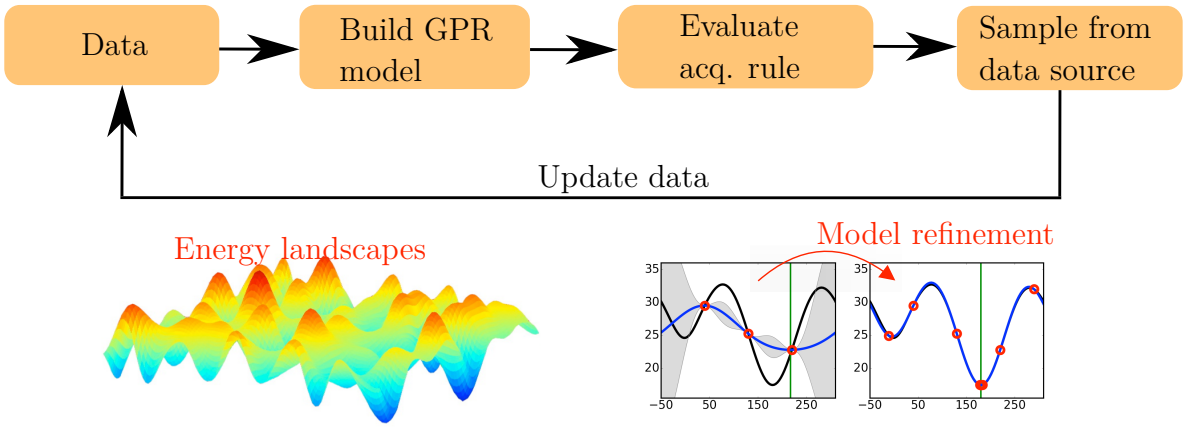


Figure 2.2: Bayesian optimization workflow. For given data, a surrogate model is fitted. The acquisition function then uses this model to predict the next sampling location. The black-box function is evaluated at this location, the data set is then updated with the new obtained sample.

repeated for a fixed sequence length, or until our computational budget is exhausted. The resulting model can then be used to make a prediction on the optimum location and value.

2.1.1 Probabilistic modelling with Gaussian Process Regression

Before introducing GPR, I define stochastic processes [27], to which Gaussian Processes (GP) belong. Let $\mathcal{T}$ be a subset of $[0, \infty)$. A set of random variables $\{Y_t\}_{t \in \mathcal{T}}$, indexed by $t \in \mathcal{T}$, is called a stochastic process. If, e.g. $\mathcal{T} = 1$, meaning the index set contains only a single element, the process would correspond to a single random variable. For a finite number of elements $\mathcal{T} = \{t_1, \ldots, t_n\}$, the stochastic process would be a random vector. Stochastic processes are generalizations of random vectors which are often used to model systems with random behavior. Without going further into mathematical details, it is important to mention that in practical applications like in natural sciences, signal processing, computer science or economics, stochastic processes usually correspond to finite dimensional joint distributions of random vectors.

One stochastic process that most physicists are familiar with is the process that models Brownian motion. This model describes the dynamics of a particle in a fluid, where it collides *randomly* with other particles. Each time, where we would measure the dynamics of a single colloidal particle over time, we would get a different trajectory. The statistics

of these different paths can be described as *sampled functions* from the so-called Wiener process [27]. Mathematically, the Wiener process W_t is characterized by

1. $W_{t=0} = 0$
2. W_t is continuous
3. W_t has independent increments
4. $W_{t_n} - W_{t_m} \sim \mathcal{N}(0, n-m)$ for $0 \leq m \leq n$,

where $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution around mean μ with width σ.

Stochastic processes, where the random variables follow a joint Gaussian distribution, are the so-called GPs. Applied to statistical learning, where GPs are used to model functions, we obtain so-called GPR models. These models are the most common choice in BO to fit a surrogate model (the posterior distribution), that reflects properties of the black-box function. Mathematically speaking, GPs are distributions over functions $f(\cdot)$ that are defined by a mean $\mu(\cdot)$ and a positive definite covariance function $k(\cdot, \cdot)$. When using GPR in BO, the input domain $\mathcal{X}$ from which the x are selected, corresponds to the input space of the black-box function, which is often spanned over space or time coordinates. The mean function $\mu(x)$ describes the expectation value of the sampled functions $f(x)$, that is, the value we would obtain if we were to calculate the average of several samples of the GP. In practice, the mean is often set to the average of the observed function values, which will be 0 after the data is normalized. The covariance function $k(x, x')$ describes properties related to the input space, such as the level of smoothness of our black-box function, or a periodicity. A common choice for the kernel is the radial basis function (RBF) kernel

$$k_{\text{RBF}}(\mathbf{x_1}, \mathbf{x_2}) = \exp\left(-\frac{1}{2}(\mathbf{x_1} - \mathbf{x_2})^\top \Theta^{-2}(\mathbf{x_1} - \mathbf{x_2})\right) ,$$

where Θ is a vector containing length scales θ_d for each dimension $d = 1, \ldots, D$. The standard periodic (STDP) kernel function is defined as

$$k_{\text{STDP}}(\mathbf{x_1}, \mathbf{x_2}) = \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\left(\frac{\sin\left(\frac{\pi}{T_d}(x_{1,d} - x_{2,d})\right)}{\theta_d}\right)^2\right]$$

where T_d is the period of dimension d. This kernel can be used when the black-box function is known to be periodic in the input domain.

In GPR, it is assumed that the function values are drawn from a multivariate Gaussian distribution. The multivariate Gaussian distribution is a generalization of the one

dimensional normal distribution to higher dimensions. For k dimensions, the distribution is defined as

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \,,$$

for a k dimensional random vector $\mathbf{x}$ with the k-dimensional mean $\boldsymbol{\mu} = (\mathbb{E}[x_1, \ldots, x_k])^T$ and the positive semi-definite $k \times k$ covariance matrix $\Sigma_{i,j} = \mathbb{E}[(x_i - \mu_i)(x_j - \mu_j)]$. The mean describes the location of the multivariate normal distribution. The elements of the symmetric covariance matrix describe variances of the individual dimensions (main diagonal) as well as covariances between the dimensions. The covariance [28] is a measure of the joint variability of two random variables x_i and x_j. Normalizing the covariance gives the linear correlation coefficient between two dimensions.

Consider a given data set with n samples $\{X, \mathbf{y}\}$, where $X = (\mathbf{x}_1, ..., \mathbf{x}_n)$ are the points in the input domain $\mathcal{X}$ (often $\mathcal{X} = \mathbb{R}^{\mathcal{D}}$, with D the input dimension) and $\mathbf{y}$ as the corresponding, possibly noisy, observed function values $\mathbf{y} = f(X) + \boldsymbol{\varepsilon}$, where the noise is modelled as $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$. The joint multivariate distribution over the observed values $\mathbf{y}$ and a predicted value y at $\mathbf{x} \in \mathcal{X}$ is

$$\begin{bmatrix} \mathbf{y} \\ y \end{bmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} K(X,X) & K(X,\mathbf{x}) \\ K(\mathbf{x},X) & k(\mathbf{x},\mathbf{x}) \end{pmatrix}\right) . \tag{2.2}$$

Here, $K(X,X)$ is the kernel matrix, containing covariances between the samples X

$$K(X,X) = \begin{pmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & \ldots & k(\mathbf{x}_1,\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & \ldots & k(\mathbf{x}_n,\mathbf{x}_n) \end{pmatrix} ,$$

where $K(X,\mathbf{x})$ contains the covariances between the samples X and the prediction location $\mathbf{x}$. We can now formulate the *conditional distribution*

$$p(y|\mathbf{x}, \{X, \mathbf{y}\}) \sim \mathcal{N}(\mu(\mathbf{x}|\{X, \mathbf{y}\}), \sigma^2(\mathbf{x}|\{X, \mathbf{y}\})$$

by conditioning the joint distribution equation (2.2) on the observed data set $\{X, \mathbf{y}\}$. With that, the results for the *posterior mean* and *posterior variance* become [29]

$$\mu(\mathbf{x}|\{X,\mathbf{y}\}) = K(\mathbf{x},X)[K(X,X) + \mathrm{diag}(\boldsymbol{\sigma}_\varepsilon^{\mathbf{2}})]^{-1}\mathbf{y} \tag{2.3}$$

$$\sigma^2(\mathbf{x}|X) = k(\mathbf{x},\mathbf{x}) - K(\mathbf{x},X)[K(X,X) + \mathrm{diag}(\boldsymbol{\sigma}_\varepsilon^2)]^{-1}K(X,\mathbf{x}) \tag{2.4}$$

$$\tag{2.5}$$

Note that the posterior variance function does not depend on the function values $\mathbf{y}$. The hyperparameters Θ of the kernel function $k(\mathbf{x}, \mathbf{x}')$ are often inferred by maximizing the

log marginal likelihood $\log p(\mathbf{y}|X)$, defined as

$$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T[K(X,X) + \mathrm{diag}(\boldsymbol{\sigma}_\varepsilon^2)]^{-1}\mathbf{y} - \frac{1}{2}\log|K(X,X) + \mathrm{diag}(\boldsymbol{\sigma}_\varepsilon^2)| - \frac{n}{2}\log 2\pi\,.$$

For each Θ, a set of hyperparameters of the regression model, the likelihood $p(X|\theta)$ assigns a probability to observe data X with a by Θ parameterized given model. Maximizing the marginal likelihood term provides the hyperparameters that are most likely to have generated the observed data, this is the so-called *maximum likelihood estimation*, a common technique of choosing the kernel hyperparameters.

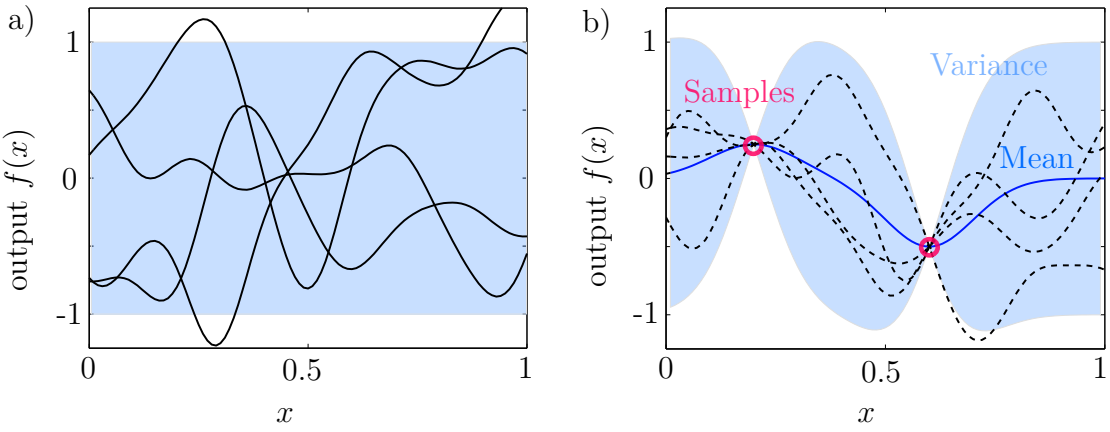


Figure 2.3: GP before and after conditioning on samples. a) Sampled functions from the GP with constant variance (blue shaded area) and constant mean (set to zero, not plotted here). This is called the *prior distribution*. b) Conditioning the GP on two samples changes the mean function (here the blue solid line). The variance at the sample location vanishes and gets largest in region where no samples are. This is called the *posterior distribution*. Parts of the figure adapted from [29].

2.1.2 Acquisition strategies

The second ingredient in the BO algorithm, besides the probabilistic model, is the use of a goal-directed sampling strategy, where the goal is to find the global minimum. This strategy is determined by the so-called AF. The AF is a function that is often chosen heuristically with little effort, i.e. a function that does not provide an optimal, but often a useful solution to the underlying problem. By design, the AF should fulfill several tasks:

- The function should be sample-efficient. This means that the BO algorithm should need as few samples of the black-box function as possible to find the global minimum.
- The sampling strategy in search space, the space over which the black-box function is minimized, should balance between exploration and exploitation. Exploration refers to the sampling of environments in search space where the variance function takes on large values, these are environments in which there are still few samples. Exploitation refers to the sampling of regions in which the posterior mean approaches low values. It is expected that the global minimum is located in these regions.

Apart from that, we would like to be able to have an AF that can be interpreted. The AF is mathematically constructed in such a way that for a given GPR model, we obtain our next sample candidate $\mathbf{x}$ by finding the global minimum of the acquisition function. Therefore, our AF should be quick to evaluate and ideally also have a gradient available, as this allows gradient-based methods to be used for efficient minimization (efficient with respect to CPU time). The following chapters provide a small set of the in the BO community established AFs, [24, 30] provide good summaries on this topic.

Exploration and Exploitation

Pure *exploration* (or respectively *exploitation*) sampling strategies can be achieved with

$$\alpha_{\text{exploration}} = \mu(\mathbf{x}) \tag{2.6}$$

$$\alpha_{\text{exploitation}} = -\sigma(\mathbf{x})\,. \tag{2.7}$$

$$\tag{2.8}$$

Those functions are usually not considered for the standard BO algorithm. The pure exploration AF would only sample regions in the search space that have a high variance. This misses the actual goal of BO: the sample-efficient determination of the global minimum. The exploitation function would sample regions where the mean function of the GPR model has low values. This strategy would, as soon as a minimum region was found, only "exploit" this minimum. The problem here is that the BO algorithm would get stuck in the first found minimum. We would then often have found only a local minimum, but we would have missed our actual goal of finding the global minimum. Ideally we would like to use a function that takes both, exploration and exploitation phases, into account.

Exploratory lower confidence bound

The Lower Confidence Bound [31] (LCB) is a popular choice for the acquisition function. Empirical tests have shown that this heuristic provides a sample efficient search

strategy [32], while also being cheap to evaluate and easy to interpret. It is defined as

$$\alpha_{\mathrm{LCB}(\mathbf{x})} = \mu(\mathbf{x}) - \kappa\sigma(\mathbf{x}) \tag{2.9}$$

where κ is the so-called exploration weight. In LCB, the first term μ *exploits* regions with low function values. The second term σ *explores* unknown regions in the search space. The weight κ can be set to favor either exploitation or exploration. It can sometimes be difficult to determine a good value for κ. Therefore, as an alternative, the weight can be determined automatically. In previous work [33], κ was determined via

$$\kappa = \sqrt{2\log_{10}(n^{(D/2)+2} \cdot \pi^2/0.3)}\,,$$

depending on the number of samples n and the search space dimension d. This is a good choice for the following reason: as the function is learned more and more precisely, the variance vanishes and the exploration is neglected. As a result, the BO algorithm might get stuck in a local minimum. With this choice of κ, exploration gets more favored at later BO steps, this is the so-called Exploratoy Lower Confidence Bound (ELCB) AF. ELCB is often easier to minimize than other acquisition functions, as it does not suffer from having large flat surfaces like other acquisition functions often do, which can be particularly challenging to minimize in high-dimensional search spaces.

Expected Improvement

The Expected Improvement (EI) function is another commonly used AF. As the name suggests, the idea is to formulate an expected improvement mathematically: Suppose we are able to draw one more sample from our black-box function. After evaluating this sample, we need to provide a solution that corresponds to the sample with the smallest observed function value from our data set. If y_* denotes the current lowest observed function value for a set of samples, EI describes, how much improvement we would expect for y_* (where improvement means a lower y value) if we would be able to include another yet unknown sample at $\mathbf{x}$. This unknown sample can be estimated using the posterior model. We can formulate improvement as

$$I(\mathbf{x}) = \max(y_* - Y, 0) \tag{2.10}$$

where Y is a random variable with $y(\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$. We can calculate the expectation for the improvement by using the posterior statistics as

$$\alpha_{\mathrm{EI}}(\mathbf{x}) = \mathbb{E}_{y\sim\mathcal{N}(\mu,\sigma)}\left[I(\mathbf{x})\right] \tag{2.11}$$

$$= \ldots \tag{2.12}$$

$$\alpha_{\mathrm{EI}}(\mathbf{x}) = (y_* - \mu(\mathbf{x}))\Phi\left(\gamma_*(\mathbf{x})\right) + \sigma(\mathbf{x})\phi\left(\gamma_*(\mathbf{x})\right) \tag{2.13}$$

$$\text{with} \quad \gamma_*(\mathbf{x}) = \frac{y_* - \mu(\mathbf{x})}{\sigma(\mathbf{x})} \tag{2.14}$$

where ϕ is the probability density function (PDF) and Φ is the cumulative density function (CDF) of the standard normal distribution. The EI is computationally inexpensive to evaluate, however, it suffers from having large flat areas at later BO iteration steps [24], making it difficult to minimize.

Entropy search based functions

The heuristic AFs presented so far have shown [24, 30] to provide sample-efficient solutions for the inference of the global minimum. So-called entropy-based acquisition functions [34–36], which originate from the field of information theory, are alternative approaches that have also proven to be sample-efficient. In contrast to the AFs from the previous chapters, entropy-based AFs are theoretically derived. The original version of the AF, so-called Entropy Search (ES) was formulated in reference [34]. In ES, the approach is to maximize the *information gain* about the global optimum *location* $\mathbf{x}_*$. Maximizing information in this context means to minimize uncertainty. The Entropy Search (ES) AF is constructed using the mutual information I between the global optimum location $\mathbf{x}_*$ and the next sample $\{\mathbf{x}, y\}$. Mutual information between two random variables quantifies how much information one would get about a random variable if the second variable had been observed. Treating our global optimum location as a random variable with an assumed distribution $p(\mathbf{x}_*|\mathcal{D})$ for it, the AF can be written as mutual information between the next sample $\{\mathbf{x}, y\}$ and $\mathbf{x}_*$:

$$\begin{aligned} \alpha_{\mathrm{ES}}(\mathbf{x}) &= I(\{\mathbf{x}, y\}; \mathbf{x}^*|\mathcal{D}) && (2.15)\\ &= \mathbb{H}(p(\mathbf{x}^*|\mathcal{D})) - \mathbb{E}[\mathbb{H}(p(\mathbf{x}^*|\{\mathbf{x}, y\} \cup \mathcal{D}))] && (2.16)\\ & && (2.17) \end{aligned}$$

Here, the expectation is calculated with respect to $p(y|\mathbf{x}, \mathcal{D})$, where $\mathcal{D}$ is the current observed data that builds our surrogate model. Note that $p(\mathbf{x}_*|\mathcal{D})$ is a D dimensional distribution, where D is the search space dimension. Consequently, the calculation of the entropy terms in equation (2.17), in which integrations must be performed, can become intractable in high dimensional search spaces.

Since the mutual information is symmetric in the input variables, the random variables can be swapped, which leads to the so-called Predictive Entropy Search (PES) [35] AF

$$\alpha_{\mathrm{PES}}(\mathbf{x}) = \mathbb{H}(p(y|\mathcal{D}, \mathbf{x})) - \mathbb{E}[\mathbb{H}(p(y|\mathcal{D}, \mathbf{x}, \mathbf{x}^*))] \tag{2.18}$$

where the expectation is calculated with respect to $p(\mathbf{x}_*|\mathcal{D})$. The advantage of PES over ES is that the left entropy term is now formulated with respect to the posterior predictive distribution $p(y|\mathcal{D}, \mathbf{x})$. Since the posterior distribution is constructed using

a GP, this entropy term can be solved in closed form, meaning that there is an exact mathematical solution for the integration. The right term must still be approximated, since $p(\mathbf{x}_*|\mathcal{D})$ occurs there again, making the acquisition function intractable for high dimensional search spaces. Nevertheless, with the PES AF, we only have to approximate one instead of two integrals, as it was the case with ES.

Empirical tests [36] have shown that using the function value y_* at the optimum instead of the location $\mathbf{x}_*$ is as good as the ES or PES AF. With this knowledge, a trick can be used to avoid the previously mentioned expensive integration. The integration still has to be calculated, but is carried out with respect to a one-dimensional probability distribution $p(y_*|\mathcal{D})$. This is the so-called Max-value Entropy Search (MES) [36] acquisition function (as it was originally formulated for a maximization problem), defined as

$$\alpha_{\text{MES}}(\mathbf{x}) = \mathbb{H}(p(y|\mathcal{D}, \mathbf{x})) - \mathbb{E}[\mathbb{H}(p(y|\mathcal{D}, \mathbf{x}, y^*))] \tag{2.19}$$

$$\approx \frac{1}{N} \sum_{y^* \in Y^*} \left[\frac{\gamma_{y^*}(\mathbf{x})\phi(\gamma_{y^*}(\mathbf{x}))}{2\Phi(\gamma_{y^*}(\mathbf{x}))} - \log(\Phi(\gamma_{y^*}(\mathbf{x}))) \right] \tag{2.20}$$

where the expectation is calculated with respect to the probability distribution of the function value $p(y_*|\mathcal{D})$. The probability in the first term in equation (2.19) is a Gaussian distribution, so the entropy $\mathbb{H}[p(x)] = \int p(x) \log p(x) dx$ can be once again calculated and solved in a closed form. The probability in the second term is a truncated Gaussian distribution. Truncated means that, given a y_* as our lowest sample value, y needs to satisfy $y > y_*$. In practice, $p(y_*|\mathcal{D})$ is approximated using the posterior model or a Gumbel distribution. Since this is a one dimensional distribution, the integral in equation (2.19) can be approximated with a sum, using only a few samples.

2.2 Extension to multi-fidelity Bayesian optimization

In this section, I will formalize the concept of MFBO, which has already been motivated in section 1.2. In MFBO [24], a Multi-task Gaussian Process Regression (MTGPR) model is used to model multiple tasks simultaneously. It is assumed that the different tasks are correlated, so the MTGPR model should be constructed in such a way to take advantage of this correlation in the predictions. In our application, we want to simulate a system using different levels of theory. This can be translated into a set of functions that describe the same target or property, but have different fidelities. The different tasks correspond to functions that describe the same objective, but have different levels of accuracy and also different evaluation costs. Our aim is to optimize the task with the highest fidelity, while using samples from lower fidelities which are less accurate, but can be much cheaper to sample. This requires the construction of a multi-task model, that

enables an information exchange between the fidelities, as well as an AF that takes also the correlation between different fidelities into account.

The original BO workflow (Figure 2.2) is therefore extended to a MFBO workflow (Figure 2.4): Now we have multiple data sources corresponding to simulators with different fidelity levels, to which we fit a *multi-fidelity GPR model*, while a *multi-fidelity AF* determines our sampling strategy.

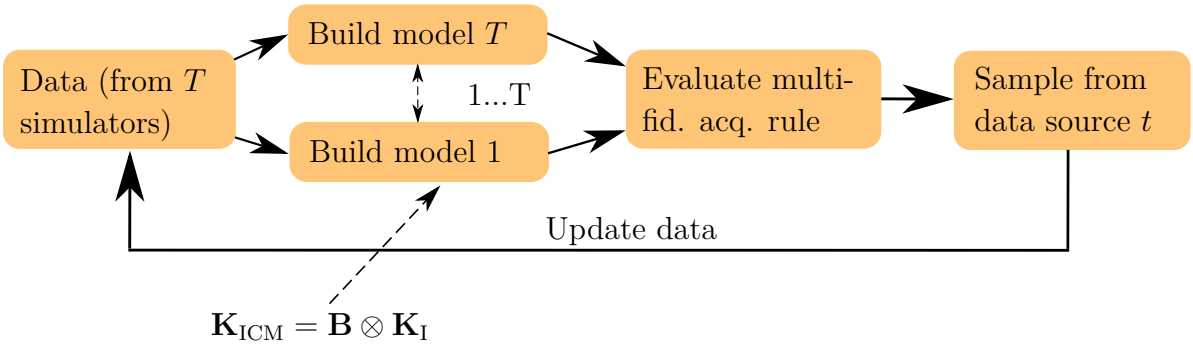


Figure 2.4: Workflow of multi-fidelity BO. The multiple data sources get modelled with a MTGPR model, introduced in section 2.2.1. In addition, a multi-fidelity acquisition rule gets introduced to sample from multiple data sources.

2.2.1 Intrinsic Model of Coregionalization

A common approach to multi-fidelity modeling is the Linear Model of Coregionalization (LMC) [3, 37]. The idea of LMC is to represent the fidelities as linear combinations of independent latent functions $u_j^q(\mathbf{x})$, with $q = 1, ..., R_q$ and $j = 1, ..., R_J$. These functions u_j^q are independent samples from a GP, and each index q corresponds to a different spatial kernel $k_I^q(\mathbf{x}, \mathbf{x}')$. The coefficients of this linear combination are related to the covariance between the fidelities, as we shall soon see.

We are going to make two assumptions that give rise to a simplified LMC model for the construction of our multi-fidelity approach. The first assumption is that our different fidelities share the same search space. This is reasonable and also useful, as it allows us to learn the spatial kernel hyperparameters using only samples from cheaper, lower fidelity sources. This also means, that we are restricting the number of spatial kernels to $R_q = 1$. With that, all fidelities use the same spatial kernel for the input space $k_I(\mathbf{x}, \mathbf{x}')$. Secondly, we assume that our MFBO kernel can be separated into a kernel that describes properties of the input space (spatial kernel K) and a kernel that describes correlations

and variances between the fidelities (task kernel B). With those assumptions, we can write our multi-fidelity model kernel as

$$k_{\text{ICM}}((\mathbf{x}, t), (\mathbf{x}', t')) = B_{t,t'} k_I(\mathbf{x}, \mathbf{x}') \,. \tag{2.21}$$

This kernel can be substituted with the kernel in equation (2.5) to extend GPR to MTGPR and obtain the so-called Intrinsic Model of Coregionalization (ICM). The ICM model can be interpreted in the following way [3, 37]: As with LMC, each fidelity is implicitly modelled as a linear combination of independent latent functions, e.g. for two fidelities (HF as high and LF as low fidelity)

$$f_{\text{HF}}(\mathbf{x}) = f_1(\mathbf{x}) = b_{11} u_1(\mathbf{x}) + b_{12} u_2(\mathbf{x}) \tag{2.22}$$
$$f_{\text{LF}}(\mathbf{x}) = f_2(\mathbf{x}) = b_{21} u_1(\mathbf{x}) + b_{22} u_2(\mathbf{x}) \tag{2.23}$$

where $u_1(\mathbf{x})$ and $u_2(\mathbf{x})$ are independent latent functions sampled from a GP that share the same spatial covariance function $k_I(\mathbf{x}, \mathbf{x}')$. Here, we can drop the superscript index for the latent functions, as we restricted the number of spatial kernels to $R_q = 1$. The weights are related to the cross-covariance between the fidelities via $B_{12} = b_{11} b_{21} + b_{12} b_{22}$. If there is a strong correlation between the fidelities, the cross-covariance B_{12} would also be large. Therefore, the models would give a large weight to the same latent function u_i. Likewise, the case of no correlation between the models would correspond to $b_{12} = b_{21} = 0$. This would be the same as modelling the fidelities independently with the same spatial kernel. Since equation (2.22) and 2.23 form a system of linear equations, we can rewrite

$$\begin{aligned} f_{\text{HF}}(\mathbf{x}) &= f_{\text{LF}}(\mathbf{x}) + u'(\mathbf{x}) \,, \text{ with} \\ u'(\mathbf{x}) &= (b_{21} - b_{11}) u_1(\mathbf{x}) + (b_{22} - b_{12}) u_2(\mathbf{x}) \,. \end{aligned}$$

Consequently, we see that the ICM is able to capture any relationship $u'(\mathbf{x})$ between the fidelities with the same smoothness as that of the input space. The parameter of the kernel matrix $\mathbf{B}$ can be treated like other hyperparameters, and is therefore also inferred by maximizing the log marginal likelihood function. We can establish positive semi-definiteness for $\mathbf{B}$ by using the Cholesky decomposition $\mathbf{B} = \mathbf{L}\mathbf{L}^T$. The rank of $\mathbf{B}$ corresponds to the number of latent functions that are used to model the fidelities, in our example $\text{rank}(\mathbf{B}) = 2$. For $\text{rank}(\mathbf{B}) = 1$, the higher fidelity model would be only a rescaled version of the lower fidelity model.

2.2.2 Transfer learning strategy

Transfer learning [38] refers to the field of research in machine learning in which information from an already solved problem is applied to a new problem. This technique is often

used in deep learning. Deep learning is based on building parameterized models that are optimized during training with labelled data to solve a regression or classification problem. Without going into further detail, the use of transfer learning there can be described as follows: An already trained model is trained on a new data set, with most of the parameters remaining fixed in the model. In this way, the model is adapted to the new related prediction task without having to train the model from scratch. If the datasets from the original training and from the new prediction task are highly correlated, this approach can reduce the training time by a very large duration. Despite the reduction in training time, very good accuracies can still be achieved, even for small datasets.

GPs are parameter-free regression models since there are no weights to be learned, contrary to, e.g., the models in deep learning. Nevertheless, we can make use of the idea of transfer learning to accelerate the optimization and reach the same global minimum prediction accuracy with less computational cost. Given a higher (target) and a lower (support) fidelity source that are linearly correlated, we can start by performing single-fidelity BO on the support fidelity. The samples resulting from the single-fidelity BO can then be used to initialize the ICM model. This is the idea from transfer learning, the reuse of a previously trained model, or, as in this case, a dataset obtained by applying BO. Since our model is not parameterized, we use the ICM model to reuse previous information by incorporating support fidelity samples in the model, obtained from the previous task. This gives us useful information about the search space from the support fidelity task before we even start BO on the target fidelity task. By continuing the BO, but only sampling the target fidelity task, it is possible to maintain the target fidelity accuracy for the global minimum inference, while requiring fewer target fidelity samples.

Before transfer learning can be applied, however, there are several issues that must be considered. Since the ICM kernel captures covariances in the input and output spaces, it has a higher computational complexity than the single-fidelity kernel. This raises the question, for which kind of configurations, e.g. what kind of fidelity levels, transfer learning is more efficient than single-fidelity BO. More expensive in this context means that it would increase the time for the minimum inference. It is also not trivial to understand, how many samples from the support fidelity task should be used for the initialization of the ICM model, as well as if those samples should actually be chosen according to a previous BO run. The number of support fidelity samples in the initialization step of transfer learning can have a significant impact on computational savings, as has been discussed in previous work [39]. Furthermore, the question can be asked whether it would be more computationally efficient to choose the samples of the support fidelity task randomly, instead of a preceding single-fidelity BO calculation. These questions were also addressed in a previous study [39] and will be answered in this research project. We can already anticipate that the most significant factors for

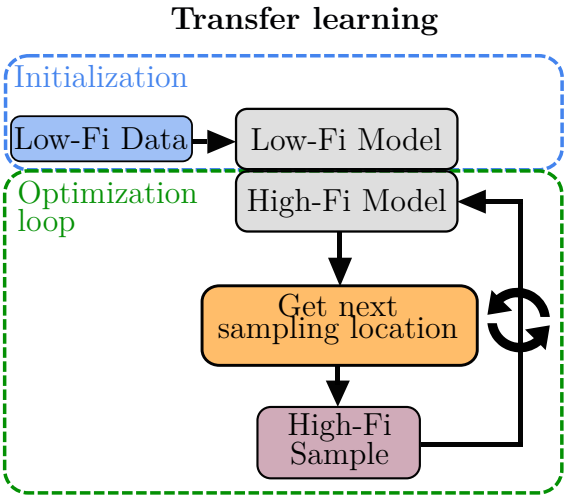


Figure 2.5: Transfer learning approach for MFBO. Support fidelity data, obtained from a previous BO run or sampled randomly, is used to initialize the ICM model. We then continue by only sampling the target fidelity source.

computational savings are the correlation and the simulator costs between the fidelity levels (costs measured in CPU time, as well as the amount of support fidelity samples). Using support fidelity samples enables us to incorporate more information about the search space, and accelerate the exploration phases of BO (Figure 2.5).

2.2.3 Multi-task learning

In a second scenario, sketched in figure 2.6, we extend our model such that all available fidelity levels are dynamically sampled throughout the BO. Even though we sample from different simulators, we are still only interested in the global minimum inference of the highest fidelity source. With the multi-fidelity learning strategy, the number of support fidelity samples no longer has to be chosen during initialization, as was the case with transfer learning. In this approach, what we refer to as *multi-task learning*, we let the MFBO algorithm determine the amount of support fidelity samples and also, at which iteration step which sources should be sampled.

A key challenge for MFBO is the design of acquisition rules. This heuristic rule should take the correlation and simulator costs into account, while maintaining cost-efficiency for the global minimum inference of the most accurate fidelity level. The multi-fidelity AF should choose not only the next sample location $\mathbf{x}$, but also the next fidelity t from which

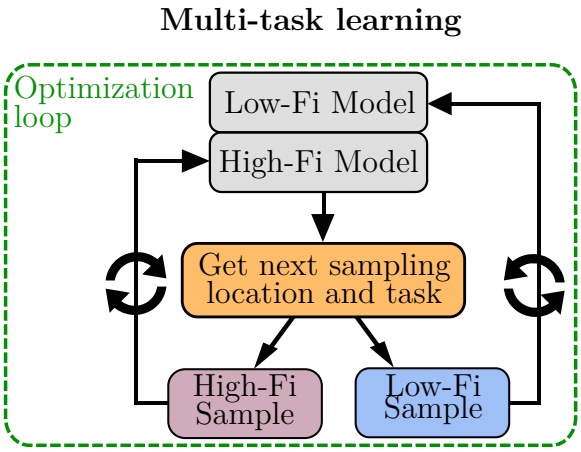


Figure 2.6: Multi-task learning approach for MFBO. We let the acquisition rule decide, at each iteration, where in search space to sample and from which source to sample from. Consequently, we can once again make use of support fidelity samples to incorporate information about the search space and reduce the cost for global minimum inference of the highest fidelity.

to calculate $f_t(\boldsymbol{x})$. As we have seen already in section 2.1.2, there are several ways to set up an acquisition strategy, even when we only have one fidelity level available. Since the acquisition rule is now supposed to determine both $\mathbf{x}$ and the task t, we can distinguish between separable and inseparable approaches, explained in more detail below.

Before we introduce the approaches that have been implemented and tested in this work, we are going to introduce a previously proposed acquisition rule based on *information gain per cost* [15, 19] $g(\mathbf{x}, t, t')$. The ICM model corresponds to a multivariate normal distribution over the extended space of input variables with $\mathbf{x} \in \mathcal{X}$ and fidelities with $t \in [1, 2, \ldots, T]$. We can establish how informative the evaluation of a fidelity level at a given location $\mathbf{x}_*$ would be, by investigating the change in the posterior variance of the target fidelity model, after conditioning on a new sample location and fidelity level t. The normal distribution at $\mathbf{x}_*$ for our target fidelity t', after conditioning on $(\mathbf{x}_*, t)$, has the posterior variance $\sigma_*(t', t') - \sigma_*(t, t')^2/\sigma_*(t, t)$. The second term here reduces the uncertainty of the target fidelity, given that we observed $(\mathbf{x}_*, t)$. In this context, uncertainty reduction can be interpreted as information gain for the target fidelity. Of course, we would get the most information about the target fidelity by always sampling the target fidelity itself. However, sampling the highest fidelity, our actual target, would also be more costly than sampling from lower fidelities. To take the trade-off between cost and information gain into account, we divide the information gain by the simulator

cost to obtain

$$g_*(t) = \frac{\sigma_*(t,t')^2}{\sigma_*(t,t)c(t)}, \tag{2.24}$$

where σ_* is the variance at fixed location $\mathbf{x}_*$, $\sigma_*(t,t')$ is the covariance between fidelity t and the target fidelity t' and $c(t)$ is the simulator cost for fidelity t. The cost can be chosen, e.g. as the evaluation duration in CPU time. This heuristic is used to determine which fidelity might be the most informative to sample next to learn about the highest fidelity level, while also taking the cost into account.

2.2.4 Multi-fidelity acquisition strategies

Separable approaches

During this work, I developed, implemented and benchmarked the behavior of several approaches. As the name suggests, separable acquisition rules make their choices for the location and fidelity level in a sequential order in separate steps. Figure 2.7 provides an overview of how these approaches are constructed and how they are different from another.

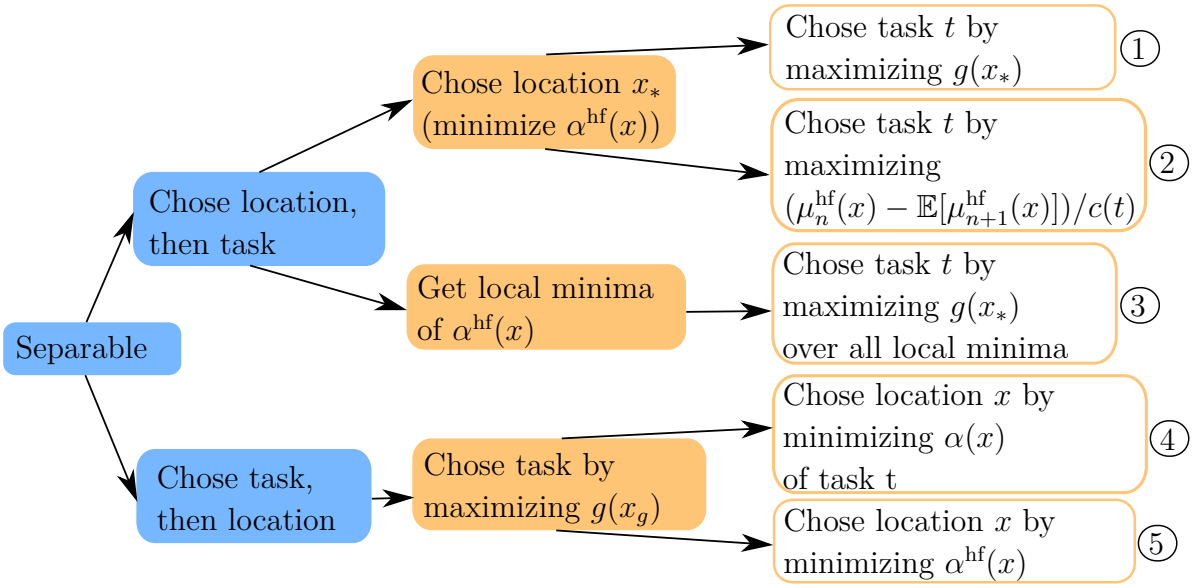


Figure 2.7: Overview of the implemented and tested separable acquisition strategies.

A1: Location then task information gain: In this approach, the location gets chosen by minimizing the acquisition function of the highest fidelity, which proposes a next

sampling candidate $\mathbf{x}_*$. For this $\mathbf{x}_*$, we maximize the information gain per cost $g_*(t)$ with respect to t, providing us information about which fidelity can be expected to give us the largest variance reduction per cost for the highest fidelity at the location $\mathbf{x}_*$. This approach has also been tested in ref. [19].

A2: Location then task posterior gain: Same as before, we receive the next sample candidate $\mathbf{x}$ by minimizing the acquisition function of the highest fidelity. For the fidelity choice, we introduce an approach based on the Knowledge Gradient (KG) [40] acquisition rule. Knowledge gradient is related to the Expected Improvement acquisition function, but it doesn't depend on the best obtained sample. If $\mu_n(\mathbf{x})$ denotes the posterior mean after n samples, $\mu_{n+1}(\mathbf{x})$ would correspond to the mean function after taking one more sample into account. KG is then defined as

$$\alpha_{\mathrm{KG}}(\mathbf{x}) = \mathbb{E}_n[\max(\mu_{n+1}) - \max(\mu_n)]$$

where the expectation is calculated using the posterior model after n samples. We reformulate this approach and chose the fidelity t that maximizes the scaled gradient of the target fidelity posterior mean

$$\mathbb{E}_n[\mu^{\mathrm{hf}}_{n+1}(\mathbf{x}) - \mu^{\mathrm{hf}}_n(\mathbf{x})]/c(t)\,,$$

where $c(t)$ corresponds to the cost of evaluating fidelity t. Adding a support fidelity sample should not change the posterior mean of the target fidelity as much as adding a target fidelity sample. Therefore, scaling by the cost takes into account a tradeoff between information of a new sample and cost of evaluating that sample.

A3: Information gain at all local minima: This approach is very similar to approach A1. Instead of only using the global minimum of the acquisition function, this approach uses all found local minima $\{\mathbf{x}_1, ..., \mathbf{x}_K\}$ of the acquisition function. For each of these K sample candidates, we maximize the information gain per cost $g_k(t)$ with respect to t. This gives us a proposal for the fidelity index t for each found local minimum $\mathbf{x}_k$. Out of all these K candidate tuples $(\mathbf{x}_k, t_k)$, we return the one which has the largest information gain per cost as our next sample candidate.

A4: Task then location: In this approach, we start by choosing the fidelity level and then pick the next sample location $\mathbf{x}$. For this, we make use of the posterior model of the target fidelity. Minimizing that model gives us the currently predicted global minimum. Once we have this prediction, we maximize the information gain per cost at this location $\mathbf{x}_g$, which suggests us the next fidelity level t. To determine the next sample location, we minimize the acquisition function α_t of the fidelity t, which provides us the next sample location $\mathbf{x}$.

A5: Task then location using the highest fidelity: This approach starts by choosing the fidelity level and then the sampling location. As in approach A4, we evaluate the

information gain per cost at the current predicted global minimum. What distinguishes this approach from the approach A4 is that, when choosing the location, we minimize the acquisition function of the model of the highest fidelity α^{hf}. In the previous approach A4, instead, the acquisition function of the model of fidelity t was minimized. Therefore, the step of choosing task and location are independent of each other: It does not matter which task is selected in the first step, the acquisition function with the target fidelity is always used for the determination of $\mathbf{x}$. Consequently, this approach would be the same as the approach A1, if we would maximize the information gain per cost there at the predicted minimum location $\mathbf{x}_g$ instead of the next sampling location $\mathbf{x}_*$.

Inseparable approaches

Inseparable approaches decide the next sampling location and fidelity level by minimizing over a joint acquisition rule. We introduce two approaches to optimize across all fidelity levels simultaneously. The first approach makes use of single-fidelity acquisition functions of all available fidelity levels t at the same time. In the second approach, we optimize over the information gain per cost, but keep $\mathbf{x}$ as a free variable instead of fixing it as we did in the separable approaches. The two inseparable approaches (Figure 2.8) are discussed in detail below.

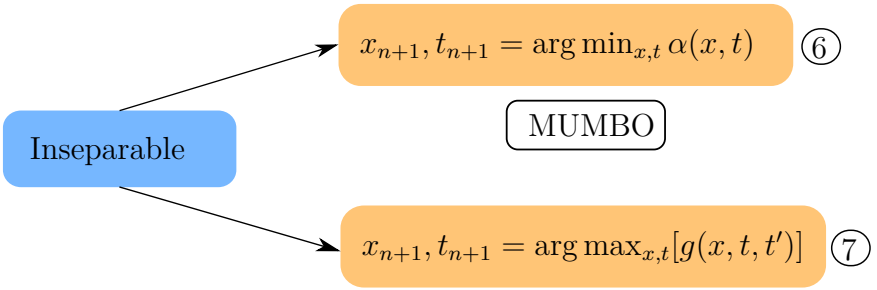


Figure 2.8: Overview of the implemented and tested inseparable acquisition strategies

A6: Inseparable For each available fidelity level t, the ICM model provides us a posterior mean $\mu_t(\mathbf{x})$ and a posterior variance $\sigma_t(\mathbf{x})$. We can use each of these posterior statistics, and formulate separate acquisition functions $\alpha_t(\mathbf{x})$ for all available fidelity levels. After rescaling the means and variances functions with the corresponding sample statistics, we can compare these acquisition functions $\alpha_t(\mathbf{x})$ with each other. This is important, as the units should be the same. For each fidelity level t, we can then optimize the acquisition function α_t, which provides us $\mathbf{x}_*^t$. Our decision on which fidelity t to

sample, will be made on

$$t = \arg\max \alpha_t(\mathbf{x}_*^t)\,.$$

We compare the optimized values for each acquisition function t, and pick the fidelity with the largest acquisition function value and the corresponding sampling location $\mathbf{x}_*^t$ as our next candidate.

A7: Inseparable information gain: In the separable approaches, information gain per cost $g_*(t)$ has been conditioned on a fixed sample location $\mathbf{x}_*$. Here, we keep the sample location as a free variable to optimize over. It is important to note that using only information gain per cost as an acquisition function is not recommended. The reason is that information gain per cost, as introduced in equation (2.24), only uses posterior variance functions. Therefore, this strategy would work very similarly to a pure exploration strategy. To avoid pure exploration, we choose an idea as described in [16]: If our maximal information gain per cost would return the task t with t being the target fidelity, we also optimize over the target fidelity acquisition function, which provides us a next sampling location $\mathbf{x}$. By this construction, we favor exploration when we sample support fidelities, while maintaining a balance between exploration and exploitation when we sample the target fidelity.

MUlti-task Max-Value Bayesian Optimization (MUMBO): MUlti-task Max-Value Bayesian Optimization (MUMBO) [17] extends the in section 2.1.2 introduced Max-value entropy search rule for multi-output models. The multi-task acquisition rule is defined as

$$\alpha_{\text{MUMBO}}(\mathbf{x}, t) = \mathbb{H}(p(y, t|\mathcal{D}, \mathbf{x}) - \mathbb{E}_{g^*}[\mathbb{H}(p(y, t|g^*, \mathcal{D}, \mathbf{x})]\,, \tag{2.25}$$

which is an extension of the MES AF (equation (2.19)). In contrast to MES, MUMBO calculates the entropies over the joint distribution $p(y, t)$ for the sample value y and the fidelity level t, instead of just the sample value $p(y)$. The second term in equation (2.25) is an expectation calculated over g_*, which is a random variable that corresponds to the global minimum value of the target fidelity level. While the second term was analytically tractable for MES, this term is intractable and gets approximated with Monte-Carlo integration, described in detail in reference [17]. MUMBO takes the correlation between the fidelities into account and is a computationally feasible multi-task AF, therefore well suited for the task of multi-fidelity BO.

In addition, I tested hybrid acquisition function approaches: The BO algorithm can keep track of the change of the global minimum prediction. If the prediction does not change over a certain amount of iterations/time, the acquisition function is switched, e.g. from ELCB to MES. Since these hybrid approaches did not turn out to be significantly better, and would also increase the complexity of our already advanced approaches (making

them more prone to errors) I decided to not continue to research these approaches any further.

It is important to clarify that the acquisition strategies introduced were chosen heuristically. This means that some strategies can be very experimental, such as approach A2. It is difficult or impossible to predict whether these approaches are useful as multi-fidelity acquisition rules before we do actual MFBO experiments with these strategies. The goal of this thesis work is to implement and test these approaches in the form of a multi-fidelity framework. This should give us insight into which multi-fidelity approaches can be more efficient than single-fidelity approaches, and how much more efficient the approaches can be. We want to reduce the computation time as much as possible while keeping the complexity of our approaches low to allow interpretability of the MFBO sampling strategies.

2.3 Simulation of atomic structures and potential energy surfaces

In this section, I introduce the three different fidelity levels used throughout the thesis. Each of these fidelity levels can be used to construct potential energy surfaces for different atomic configurations. The potential energy surface is the function that I previously denoted as "black-box function" in BO. I will start by explaining what I actually mean by a fidelity level, which requires me to give a very brief introduction to the field of modern simulation methods used in physics.

Force field (FF) [41, 42] simulations are based on classical physics and ignore quantum mechanical phenomena for the calculation of energies and forces. In FF, molecules are treated as spheres that are held together by springs. The potential energy can then be parameterized as a field by writing the energy as a sum of the bonding energy (stretching the springs), bending energy, dihedral energy and further non-bonding terms. It is important to note that FFs are carefully parameterized against results from more accurate quantum mechanical calculations, which I will introduce in the next paragraph. This ensures that the results of the FF calculations are very accurate, providing good approximations to the quantum mechanical calculations, while they can be calculated very quickly. FFs often provide reasonable atomic configurations generated by relaxation, where the atomic positions are optimized for minimum energy. This is also the idea of transfer learning presented earlier: Incorporating samples from a fidelity with lower accuracy, here force fields, for the initialization of a model to accelerate the finding of the energy minimum at a higher fidelity. I will denote the force field based calculations

used throughout this thesis as samples from a Low Fidelity (LF) simulator. For the calculation of this fidelity, I used the `AMBER18` [43] software package.

As my second level of fidelity, I use the *density functional theory* (DFT) [42] approach. The DFT formalism is a well established approach for applications in materials science and chemistry [44], providing an efficient way to study structures and reactions in atomic systems. For many systems, some properties, such as the ground state (the stationary state with the lowest energy of a quantum mechanical system) can be described using an electron density distribution alone [45]. With this, we can avoid specifying the many-body three dimensional wave function Ψ of the electron system. DFT calculations are built up on the Hohenberg-Kohn theorems, stating that the ground-state properties of a molecule are described by the electron density function and that a candidate density function must be equal or greater than the 'true energy'. In the DFT approach, energies of a system are formulated as an energy for a system with non-interacting electrons and a deviation that is described by an exchange-correlation functional. This is the so-called Kohn-Sham theorem [46]. The contribution of this functional can not be calculated exactly, approximating this functional is a major problem approached in DFT. Various approaches for this approximation exist, an overview can be found in references [44] and [42]. For my application, I will use the `FHI-AIMS` [47] software for DFT energy calculations with a PBE-exchange-correlation functional. Samples from this simulator are going to be denoted as High Fidelity (HF).

The most accurate fidelity level is built upon *ab initio quantum chemistry* [42] theory, I will denote this level as Ultra High Fidelity (UHF). This theory is, like DFT, built up on solving the Schrödinger equation by using approximations. The type of approximation determines the level of accuracy of the solution. The simplest approximation is the so-called Hartree method, where the wave function Ψ is constructed as a product of single particle orbitals. In an extension to this, the so-called Hartree-Fock method, the wave function is formed as a Slater determinant [48], describing occupied spin orbitals. These orbitals have a spatial contribution, often written as linear combinations of basis functions for atomic orbitals, as well as spin contributions. With this approach, the wave function is anti-symmetric, such that the Pauli-exclusion principle [49] is satisfied. The Hartree-Fock method fails at treating electron correlations, since in this approach, the electrons are treated as they would move in an electrostatic field that is created by the other electrons. More accurate approaches, so-called post Hartree-Fock methods, fix this by taking interaction contributions between the electrons into account.

For my applications, I employed a *coupled cluster* (CC) [42] simulator. In CC, the wave function can be rewritten as a sum of the ground state wave function Ψ_0 plus

contributions from excited electron states

$$\Psi = \left(1 + \hat{T} + \frac{\hat{T}^2}{2!} + \frac{\hat{T}^3}{3!} + ...\right)\Psi_0 = e^{\hat{T}}\Psi_0\,,$$

with $\hat{T} = \hat{T}_1 + \hat{T}_2 + ...$, where $\hat{T}_i$ are excitation operators that describe excitations of electrons into virtual unoccupied orbitals. Therefore, in contrast to the simpler Hartree-Fock approach, contributions from unoccupied orbitals are taken into account for the calculation of the energy. Depending on the highest excitation levels we include, we receive different levels of accuracy: Coupled cluster doubles (CCD), coupled cluster singles and doubles (CCSD) or coupled cluster singles, doubles and triples (CCDT), corresponding to

$$\begin{aligned}\hat{T}_{\mathrm{CCD}} &= e^{\hat{T}_2}\Psi_0\\ \hat{T}_{\mathrm{CCSD}} &= e^{(\hat{T}_1+\hat{T}_2)}\Psi_0\\ \hat{T}_{\mathrm{CCSDT}} &= e^{(\hat{T}_1+\hat{T}_2+\hat{T}_3)}\Psi_0\,.\end{aligned}$$

I used CCSD(T) in my simulations: Instead of calculating the very expensive CCSDT approach, contributions of the triple excitations are considered via a perturbation approach. For my calculations, I used the `Gaussian16` [50] simulation software, which enables me to calculate energies with the CCSD(T) approach.

2.4 BOSS: Active machine learning framework for structure search

To perform the BO experiments described in the previous chapters, I used the BOSS library. This theory chapter describes which tasks BOSS was developed for, while a following methodology chapter will describe how BOSS is implemented and what functionalities it offers.

BOSS [9] is a general purpose implementation of BO in Python with special support for applications in materials science. It is designed to learn surrogate models in a sample-efficient way and can be used without the requirement of having profound knowledge on GPR or BO. BOSS allows for easy combination of computational simulators and state-of-the-art probabilistic machine learning approaches. A typical application of BOSS is to use it for the optimization of an energy surface. Atomistic simulators map the structure of a material to an intrinsic energy, where low energy configurations indicate stable materials. BOSS applies a BO routine, by treating the atomistic simulator as the

black-box function, to find stable structures that occupy such low energy states. The process for that is summarized in algorithm 1. There are many other BO libraries such

Algorithm 1 Bayesian optimization structure search pseudo-code

Initial dataset $\mathcal{D}$
repeat
  $x \leftarrow \arg\min_x \alpha(x)$ ▷ Optimize acquisition function
  $y \leftarrow f(x)$ ▷ Evaluate simulator at x
  $\mathcal{D} \leftarrow \mathcal{D} \cup (x, y)$ ▷ Update dataset with new sample
  Refit surrogate model ▷ Maximize likelihood for updated data set
until Computational budget exhausted
return $\mathcal{D}$, predicted minimum, (optional) post-processing results

as Emukit or BoTorch. BOSS provides extra support for materials science through the following set of features:

- Restart capability, which is important for calculations on high-performance computation platforms (HPC), as they are often used for simulations in materials science. In the present work, the restart capability was used for BOSS calculations with the computationally expensive UHF simulations.
- BOSS provides well-tested default choices for the parameters of prior distributions, which is often required to be manually chosen in BO workflows. In addition, it supports to set parameter for the distributions manually.
- BOSS gives the user easy access to a variety of acquisition and kernel functions.
- Another useful feature is the post-processing routine that BOSS provides. Using this routine is quite convenient to analyze obtained optimization results, or to detect possible errors or bugs in the simulator setup. In addition, the post-processing routine can also be used for the analysis of local minima of the black-box function.

3 Multi-fidelity computational implementation

In this chapter, I describe the implementation work that was required to incorporate and test the multi-fidelity approaches in BOSS. In section 3.1, I introduce the architecture and implementation of BOSS in more detail. Section 3.2 reviews the available multi-fidelity functionalities in BOSS. I will discuss 1) an extension of the multi-fidelity functionalities that I added in section 3.3, 2) code validation in section 3.4, 3) how I measure computational savings and formulate my research objectives in section 3.5, and finally 4) experimental design in section 3.6.

3.1 Computational implementation of BOSS

BOSS is a Python optimization library which can be used to efficiently minimize black-box functions. In the following, I will discuss the implementation of BOSS, as a rough understanding of the structure of the library is needed to understand coding work for the multi-fidelity extensions. BOSS builds on top of the existing `GPy` library [51]. `GPy` is a framework that provides functionalities to use GPR with Python. On an abstract level, BOSS can be seen as a wrapper which combines the GP functionalities of `GPy` with additional methods to enable the use of BO. While `GPy` is a versatile library that provides a lot of functionalities and support for more advanced GPR methods such as noisy, sparse or multi-task models, it can be difficult to use for non-domain experts due to this flexibility and rather advanced documentation. BOSS solves this issue by constructing `GPy` models with a user-defined settings file, making the usage of BO straight forward.

The BOSS library is flexibly designed so that it is possible to easily set up BO algorithms with desired properties through a wide range of options. Alternatively, the BO setup can also be determined automatically by BOSS depending on the dimension of the search space or other expected properties of the black-box function. BOSS offers a wide range of ready to use kernel functions with according prior distributions for the hyperparameters. Functionalities regarding the BO algorithm are concluded by the availability of a choice of different AFs that determine the sampling strategy. During the optimization, BOSS

keeps track of metrics such as the global minimum prediction or kernel hyperparameters. Further functions are available to automatically evaluate the results of the BO algorithm. This includes post processing techniques that can be used to visualize the model at every iteration step, sampling locations, hyperparameter changes and metrics which track convergence behavior. If needed, a user can also make use of available local minima analysis functionalities.

Figure 3.1 visualizes the internal structure of the BOSS framework. The filled boxes in this figure correspond to Python classes. Unfilled boxes contain utility functions that are used throughout the optimization pipeline.

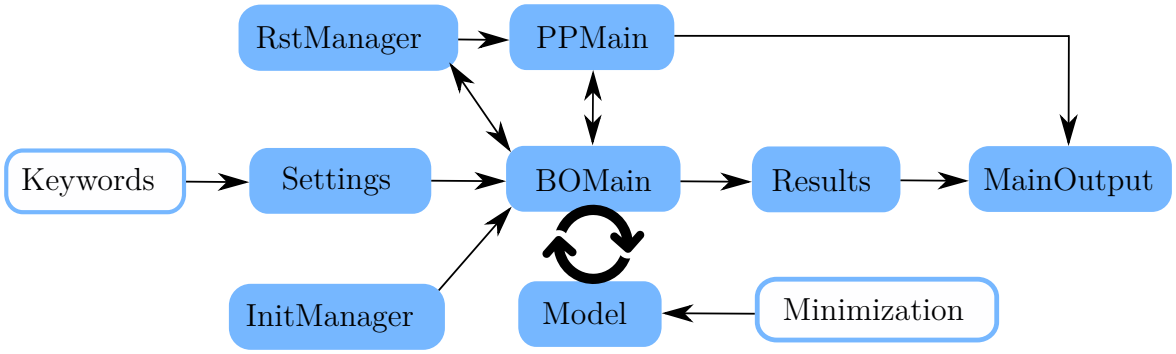


Figure 3.1: BOSS' internal structure visualized. BOSS consists of several classes, located in different modules. Arrows in this figure indicate information flow, e.g. the Settings class uses functions from the Keywords module and provides information to start the BO routine, which is handled in the BOMain class.

In the following I am going to list the classes implemented in BOSS and their corresponding purpose, for further information I refer to the official homepage of BOSS:

- **BOMain**: Initializes other objects and starts the main optimization loop
- **Model**: Creates, fits or optimizes a GP model
- **Results**: Stores optimization results
- **Settings**: Handles keywords and shares them between objects
- **MainOutput**: Writes main output files (.out)
- **RstManager**: Handles restart files (.rst)
- **PPMain**: Applies (optional) post-processing, such as plotting and local minima search

- **InitManager**: Creates initial points for the model

The above listed classes access additional utility functions which are implemented in the following modules:

- **Keywords**: Stores keywords that can be used within BOSS
- **Minimization**: Utilities to minimize surrogate models

BOSS is set up in a modular structure, in agreement with the "Pythonic approach" of utilizing features of the Python language to write clean, reusable and maintainable code. The modular structure of the library makes it possible for multiple developers to work and develop features independently. The post-processing module provides convergence analysis by visualizing the global minimum prediction as well as the sample location at each BO iteration. In addition, the module can be used to plot the posterior model and the kernel hyperparameter over iteration.

3.2 Multi-fidelity BOSS

The basic multi-task functionality was integrated into BOSS during 2019-2021 by N. Sten [39] and U. Remes. During that work, the ICM model as well as prior distributions for the ICM model hyperparameters were implemented. In addition, they introduced parameters with which the simulator costs can be handled, which is required for multi-task learning approaches. Finally, a multi-fidelity sampling strategy was implemented, referred to in this thesis as multi-fidelity approach A1, with which the ICM model can be used in the MFBO algorithm. Figure 3.2 presents the application of multi-fidelity BOSS to optimize the multi-fidelity Forrester function [52] via a transfer learning approach. Note that this approach provides a very good prediction (blue curve) for the true function (black curve) at the higher fidelity level, after acquiring only $2-4$ data points of the target fidelity function, by including support fidelity samples in the ICM model.

For the objectives of this thesis work, the functionalities for multi-fidelity approaches are extended according to the research objectives (Section 3.5) and moved to a separate module. This, and further implementation work done in BOSS, are discussed in the following section 3.3.

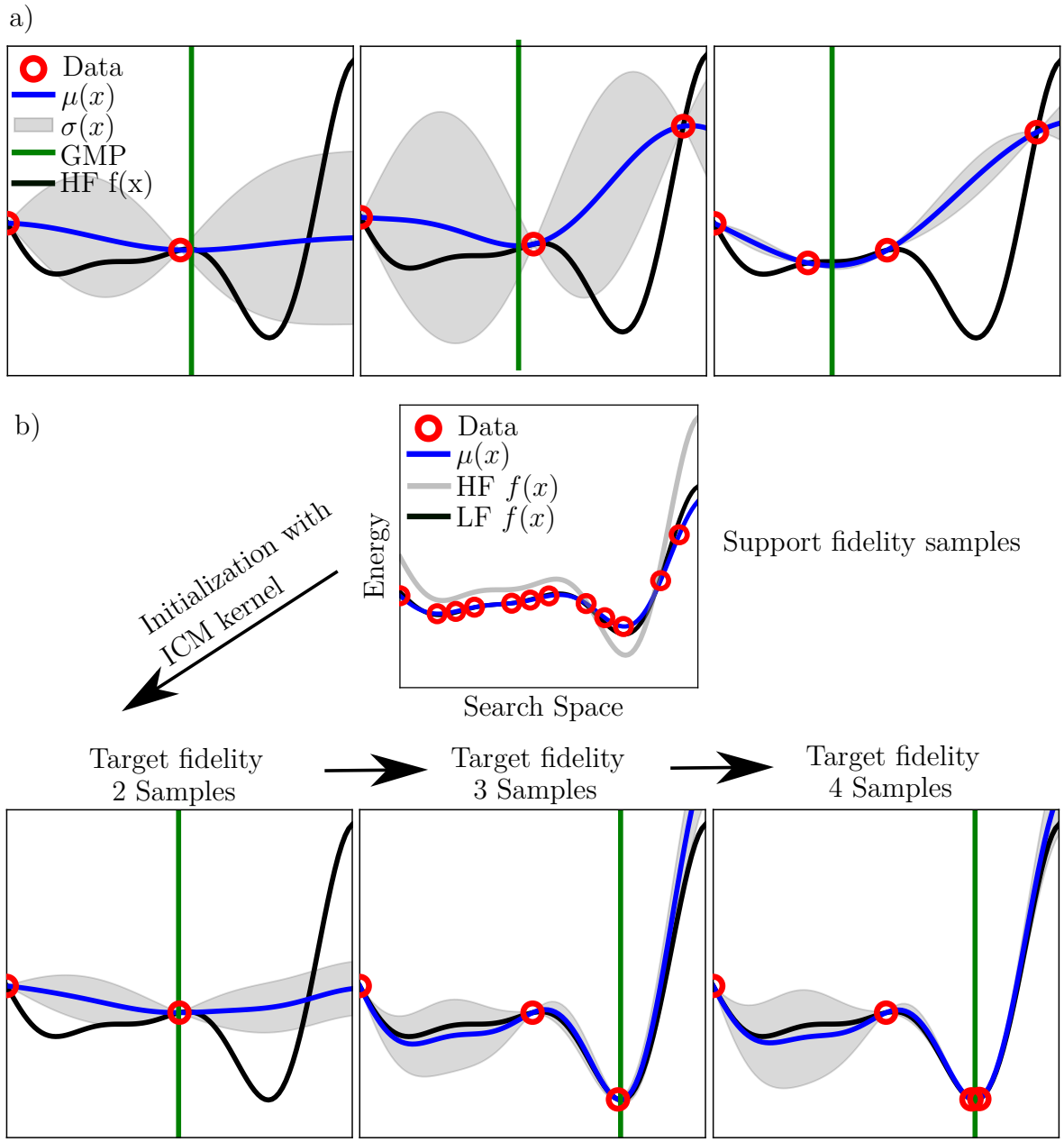


Figure 3.2: a) Single-fidelity BOSS applied to the Forrester function $f(x)$, b) Transfer learning approach applied to the multi-fidelity Forrester function $f(x)$. Data from a lower-fidelity version of the support fidelity function is used to initialize an ICM model, with which the target fidelity Forrester function's global minimum can be approximated well after only 3 sample acquisitions.

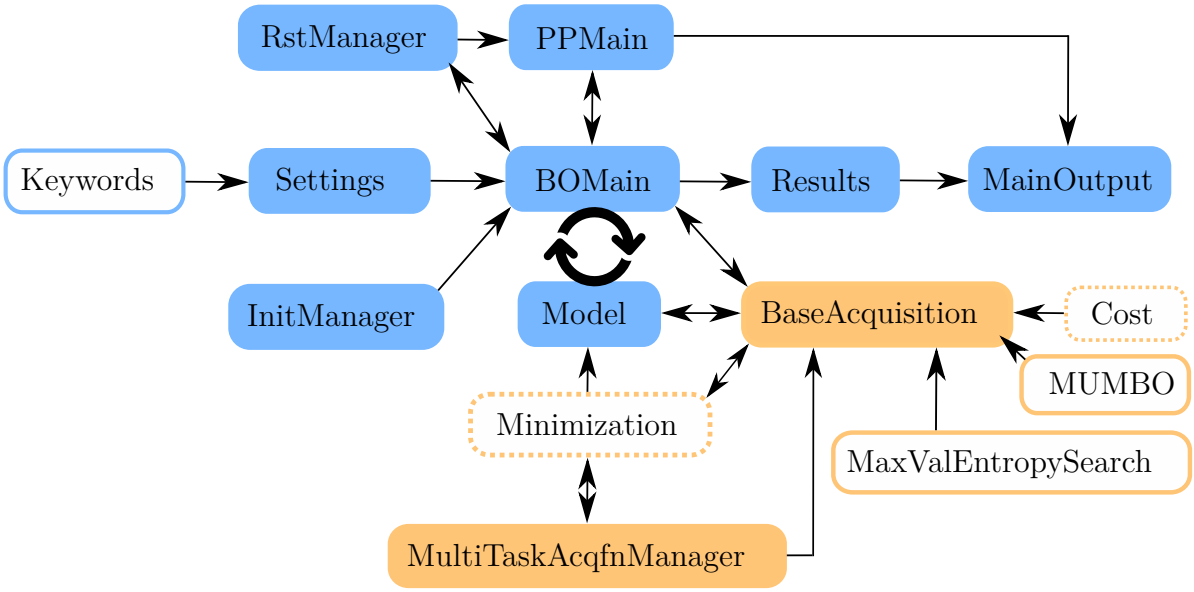


Figure 3.3: New features added (orange) throughout the work of this thesis. BOSS now has a separate Python class that handles the AFs. This includes methods for the initialization, evaluation (with or without gradient), minimization and testing of AFs. In addition, I added two entropy search-based AFs. The `MultiTaskAcqfnManager` module manages multi-fidelity approaches. Dashed box frames correspond to partial contributions.

3.3 Extending the multi-fidelity framework of BOSS

Besides the actual research work, I did implementation work on BOSS (Figure 3.3). On the one hand, this was needed to add multi-fidelity approaches to BOSS and on the other hand, the maintainability of the BOSS library was improved. My implementation work can be divided into the following steps:

- **Unifying the existing AF modules into an AF package:** BOSS had a compact framework for AF, as only single-fidelity AF were needed at that time. I rewrote the existing AF functionalities in the form of a new code package, which was a logical step and crucial for maintainability purposes. With the new structure, methods regarding the initialization and minimization of the AF are separated from the BO main module, which makes it easier to work on these parts of the package independently.
- **Adding a base template to consolidate functionalities for the AFs:** For

more standardization, I created a base class template called `BaseAcquisition` which specifies methods for the AF that are required during BO. With this, upcoming AFs can be implemented following this template. This also ensures, that methods for the evaluation (with and without gradient) and minimization are available for new AFs. The added AF code package also makes it easier to add methods for the multi-fidelity approaches, which was required at a later stage of this thesis work. Moving the minimization part of the AF to `BaseAcquisition` helped to implement and use gradient-free AFs, such as the entropy based approaches. The `BaseAcquisition` template class also makes it easier to add new functionalities, such as cost functions. Cost functions treat AFs as abstract objects and can be used as function wrappers with the objective of favoring or avoiding certain regions in search space. I added unit tests for this new framework, where all functionalities of the acquisition classes are tested for different AF choices.

- **Adding a class for multi-task acquisition strategies:** The new base structure will also simplify future implementation work, such as the incorporation of human expert knowledge for the BO guidance of real experimental setups in multi-task approaches, or adding new types of cost functions. Using the base template, I implemented a new class, the `MultiTaskAcqfnManager`, to keep MFBO acquisition strategies separate from single-task approaches. These functionalities can then be enhanced and combined with new methods, to use all seven presented multi-fidelity approaches from section 2.2.3.

These extensions to BOSS make it easier to add the new AFs. I also implemented a method that accelerates the minimization time of AFs and is more reliable in finding the correct global minimum:

- **Adding entropy AFs:** Using the `BaseAcquisition` class framework, I added two new types of AFs to BOSS: the in section 2.1.2 introduced MES and MUMBO, a multi-fidelity version of MES. Both AF make use of a gradient-free minimizer, since they don't have access to a closed-form gradient, meaning that there is no exact mathematical solution for the gradient available.

- **Improving the AF minimization algorithm:** BOSS applies the in `SciPy` [53] implemented L-BFGS [54] minimization algorithm to minimize the surrogate model and the AFs. L-BFGS is a quasi-Newton optimization technique that requires a starting point located in the search space. Starting from that point, the gradient signal gets iteratively reduced during the search for the minimum. The minimization routine is repeated multiple times from different starting points. The reason for this is that the minimum prediction often ends up in a local minimum. By repeating the minimization algorithm from different starting points in search space, the chance of

finding the global minimum is increased. In higher dimensional search spaces, more local minima are to be expected. To take this effect into account, in the original implementation BOSS started the minimizer from n randomly chosen points. Here, the number n is a function that depends on the search space dimensionality. In all cases, the number of random points is chosen between 10 and 100. This large range takes into account optimization problems with both low and high search space dimensions.

Nevertheless, this approach can potentially fail to find the global minimum. AFs tend to have flat regions at later BO iterations, making minimization more difficult: Optimal solutions would be found in very narrow regions in phase space, and random minimizer initializations might miss this region. This is because the black-box function gets better approximated by the model at later stages. Some AFs, such as EI, are based on sampling regions where we can expect improvement in terms of lower function values. However, as the black-box function is learned better and better, EI expects no further improvement over most regions in the search space, resulting in a flat surface.

During my implementation work, I tested a new initialization strategy of the minimizer, a technique that is also used in Emukit, a Python toolkit for decision-making under uncertainty [55]. Instead of choosing the starting points randomly, the AF is evaluated at around 10^3 random locations. For most AFs, the computation time to evaluate the AF at 10^3 random locations can be neglected in comparison to the time for a single L-BFGS minimization routine. These evaluations are then assigned a score, according to their AF value. The L-BFGS minimizer is then started using the $1-3$ points with the lowest score value. With that, the global minimum is found more reliably than with the previous initialization. This is also a better choice than choosing $10-100$ random starting points, as the minimization routine is only called $1-3$ times. Instead of using several starting points and therefore repeating the minimization several times, the new initialization starts only from the most promising (in terms of low function values) regions. The result is that this new initialization strategy also drastically reduces the time to find the global minimum.

3.4 Code validation

Code validation is used to check whether the implementation delivers the desired result, so it can be described as the process of checking that the code is correct. Validation methods are supposed to help to eliminate possible errors and programming bugs in the

code. In previous work [39], the multi-fidelity implementation of BOSS has already been successfully validated and tested with atomistic simulators. I'm going to build on top of this work by adding another fidelity level that is more accurate and has a significantly larger simulation cost, which will further validate the previous code implementation.

To further validate individual parts of the code, such as the added base acquisition template, I added unit tests to BOSS. Unit tests are automated tests that test individual part of the code together with control data to check whether parts of the code work as intended. For instance, in BO, the different AFs can be set up using given data and checked whether the evaluation and minimization of the AF would deliver the expected result by comparison with provided control data. I added unit tests for the AF template as well as for individual AFs.

To validate the code during the multi-fidelity implementation work, I had to run MFBO calculations. Consider the case where multi-fidelity BOSS would use a simulator which has a very long evaluation time. Checking whether all seven multi-fidelity approaches are correctly implemented would take an unfeasible amount of time. To avoid this waiting time, I created simulator models. These correspond to `GPy` GPR models that are trained on real simulator data. I trained a separate GPR model for available data from each fidelity level (the origin of this data is discussed in section 4.2), using 100 data points that I obtained from 2D single-fidelity baseline experiments. By using these simulators instead of the real simulators, the code can be checked for correctness more quickly, as the time required to evaluate a simulator is negligible compared to the overall BO iteration time.

3.5 Quantifying the advantage of multi-fidelity sampling strategies

3.5.1 Measuring computational savings

The aim of this research work is to compare multi-fidelity approaches with single-fidelity approaches. MFBO has, like single-fidelity BO, the objective to determine the global minimum sample-efficient, or put differently, *computationally* efficient. We aim to learn about both the accuracy and the efficiency of multi-fidelity approaches.

Accuracy - reliability of finding the correct global minimum

Accuracy quantifies the ability of an approach to be able to find the global minimum at all. All multi-fidelity approaches are directly benchmarked against single-fidelity experiments. The single-fidelity experiments provide us information about how many BO iterations and how much CPU time is required on average for the correct global minimum prediction, and also where the global minimum is located. Since we know how many computational resources were needed for the single-fidelity optimization, we can provide the same computational resources plus additional resources as a safety margin to the multi-fidelity approaches. If a multi-fidelity approach still can not find the correct global minimum after using these provided resources, it will be assigned a bad accuracy. Accuracy can thus be introduced as a binary label for experiments: Good for a successfully found global minimum within a resource budget, bad when the global minimum is not found. The results of BO depend on the initialization, by repeating BO with a different initialization and averaging the results, the reliability of finding the correct global minimum of a multi-fidelity approach can be quantified. Multi-fidelity approaches that fail to find the global minimum despite the availability of a sufficient amount of computational resources should be avoided.

Efficiency - required resources for global minimum detection

Efficiency refers to how fast an approach finds the global minimum, and is therefore related to CPU time and the number of BO iterations it takes to predict the correct global minimum. Given that the exact location of the global minimum and its corresponding value are known from previous experiments, I can calculate the deviation of the current global minimum prediction from the correct global minimum value at each iteration step. This deviation, plotted over iteration, is also called *regret* in the BO literature (see e.g. reference [36]). Ideally, the regret converges close to zero as the number of samples increases. To quantify the convergence time or efficiency of a single BO experiment, I consider the number of iterations required and how much CPU time is required for the regret to subsequently stay within a given tolerance interval (Figure 3.4 a)).

BOSS experiments are initialized with two starting points before the actual BO routine starts. These starting points are often randomly chosen points in the search space. Randomly chosen informative starting points could help to find the global minimum earlier, while for uninformative starting points, the time could be longer than on average. To take such random effects into account, I repeat the BO experiments with a random set of initial points, which results in different convergence times for each initialization. We are interested in the *expected* computational savings, and therefore I calculated summary

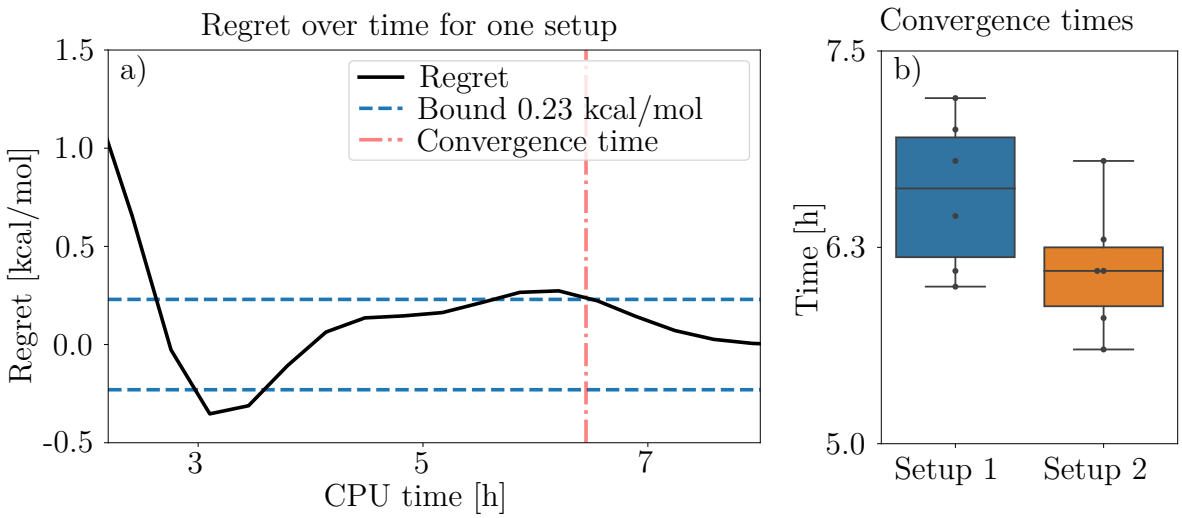


Figure 3.4: Measuring the success of the MFBO strategies in CPU time. a) Plot of the regret (deviation of the global minimum prediction to the true global minimum) over CPU time. Once the regret stays within a pre-defined tolerance, the BO run is considered converged. b) By repeating the experiments from a) we obtain statistics for the convergence times of the two setups. As indicated by the box plots, setup 2 has a lower convergence time median and a smaller variance than setup 1, and would therefore be considered a more efficient approach.

statistics for these convergence times. These statistics can then be used to determine the efficiency of the approaches (Figure 3.4 b)).

In BOSS, the most significant contributions to the computation time of a *single* BO iteration result from

- the minimization of the AF,
- refitting the hyperparameter of the model by maximizing the log marginal likelihood, and
- the minimization of the posterior model to keep track of the global minimum prediction.

Each of the listed items require the computation of an inverted $n \times n$ kernel matrix for n given training samples. This provides a bottleneck, as the computational complexity of matrix inversion grows as $\mathcal{O}(n^3)$.

Using the ICM model in MFBO increases the computational cost even further, as additional contributions to the computation time at each BO iteration arise from

- the minimization of a multi-fidelity AF,
- the larger kernel matrix, for t fidelity levels and n training samples, the kernel matrix has the size $tn \times tn$, and
- additional hyperparameters for the covariance between the fidelities that have to be fitted at each iteration, again by using the log marginal likelihood.

It must also be taken into account that the CPU time of a single-fidelity BO iteration can not directly be compared to the CPU time of a MFBO iteration. A subset of the iterations in MFBO evaluates the support fidelity level, and has therefore a much faster acquisition time. Therefore, it is also useful to monitor the number of BO iterations of the target fidelity level.

3.5.2 Research objectives of multi-fidelity Bayesian optimization

For both the transfer learning experiments and the multi-task learning experiments, we are interested in answering the questions

1. If the global minimum is found and if so, how many BO iterations and CPU time it required with respect to the single-fidelity baseline experiment.
2. If using the ICM model has the potential to reduce both the BO iterations and the CPU time to reach convergence with respect to the single-fidelity baseline experiment.
3. What kind of fidelity combinations are most efficient, and if this can be quantified by the correlation and the cost difference between the simulators.

In transfer learning experiments, we also want to learn how the convergence statistics change with the number of support fidelity initialization points. I will also directly compare the transfer learning and multi-task learning performances. One of the objectives of this research is to find out whether one of these two methods is superior in terms of computational savings over the other. If this is not the case, it would be helpful to know if conclusions can be drawn for which scenarios one of the two multi-task approaches might be the better one.

I will not only use the CPU time as a convergence measure (Figure 3.4), but also the number of BO iterations required to reach convergence. The number of BO iterations refers to the number of the target fidelity samples, excluding the samples from the support fidelity simulator. It would be interesting to see if some multi-fidelity approaches applied to certain experiments reduce the number of required BO iterations while

increasing the overall CPU time. To answer these questions, I will benchmark all multi-fidelity approaches against the results of baseline runs. A baseline run corresponds to a standard single-fidelity setup, where no lower fidelity is used as support. This will help me to conclude how many computational savings can be expected from the different approaches.

3.6 Experimental design

This chapter is introducing the structure search test system I used throughout this thesis to establish benchmarks for the multi-fidelity approaches. I will also motivate and describe the experimental designs I used for this system, after explaining in the previous chapter how I measure the efficiency of an experiment.

A common structure search problem is the problem of finding conformers (Section 1.1). Specifically, I applied conformer search to the amino acid alanine $C_3H_7NO_2$. Alanine is an ideal candidate for this, as it is a small and well studied molecule with 13 known local minima on the potential energy surface [56]. In addition, it has been used in previous BOSS applications [11, 15, 39]. This means that we know the global minimum of alanine, so all the approaches in this thesis are to be tested to see if they predict the same global minimum. The configuration space for this test system is spanned by four dihedral angles (Figure 3.5). Fixing two out of the four dihedrals allows me to test both a 2D and a 4D optimization problem. This allows me to test how the BO algorithm depends on the configuration space dimension.

To perform the multi-fidelity experiments, I used the high performance computing platform Puhti. Puhti is a supercomputer managed by Finland's' CSC – IT center for science Ltd. that allows its users to perform sequential and parallel calculations. It consists of 700 CPU nodes, each having two Intel Xeon Gold 6230 processors with 20 cores per processor. I use Puhti for all experiments with the in section 2.3 introduced atomistic simulators.

3.6.1 General settings for the alanine system

Figure 3.5 a) shows a sketch of the alanine molecule and the degrees of freedom that we optimize. In b), average acquisition times from all three fidelity levels (Section 2.3) are shown. The acquisition timing data were obtained from a correlation study that is introduced in section 4.1. Each simulator was calculating 100 different alanine configurations, the figure contains the average and the standard deviation of the timings. Because there

is such a big difference between the simulator costs, this is an ideal test system for our multi-fidelity approaches.

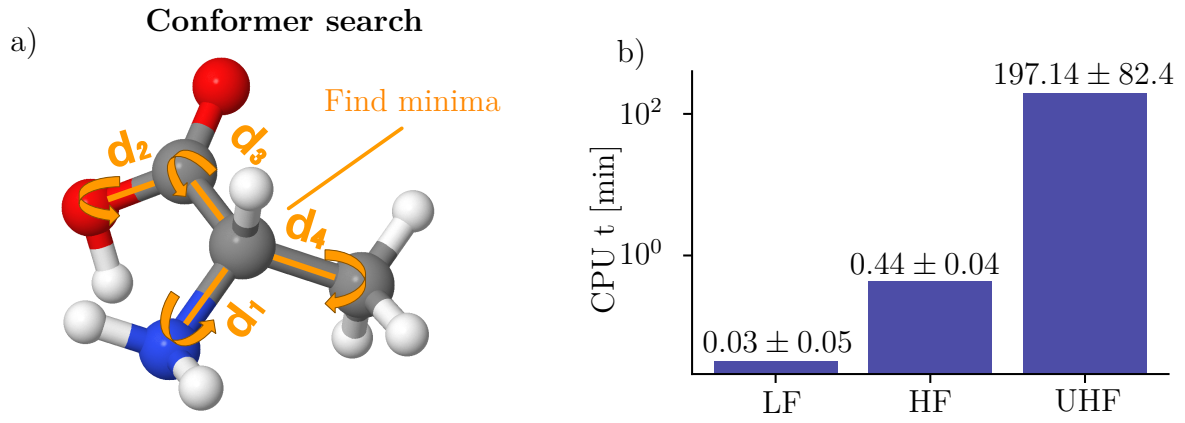


Figure 3.5: a) Alanine conformer search. In this optimization task, the amino (NH_2), methyl (CH_3) and carbonyl (C=O) groups are fixed. The bond lengths and angles are kept fixed to their average values, such that the search space is spanned by four dihedral angles $d_1 - d_4$. b) Acquisition times for the simulators. The mean and standard deviations are shown above the bars. Note that the CPU time axis is in log scale.

In single-fidelity BOSS, a user has the option to choose parameters with which the hyper-parameter prior distributions can be set. A common choice is the Gamma distribution

$$x \sim \text{Gamma}(\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x} \text{ with } \quad x, \alpha, \beta > 0\,.$$

Here, the parameters α and β determine the shape of the distribution, $\Gamma(\alpha)$ is a constant. I chose the same parameters for α and β throughout this thesis, using the recommendations for the alanine system from previous work, where the parameters for multi-fidelity BOSS have been chosen such that they reproduce results from single-fidelity BOSS [39]. There, the Gamma prior distributions were parameterized with

$$\sigma_\alpha = 2, \sigma_\beta = \frac{2}{\hat{A}^2} \quad \text{and}$$
$$l_\alpha = 3.3678, l_\beta = 9.0204\,.$$

l_i denotes the length scale prior parameters and σ_i denotes the variance prior parameters, while $\hat{A}$ is the expected amplitude of the output parameters. $\hat{A}$ is determined using the correlation study in section 4.1. I used the standard periodic kernel with a periodicity of 2π for dihedral $d_1 - d_3$ and a periodicity of $2\pi/3$ for d_4. The 2D system has fixed two out of the four dihedrals to $d_3 = 60$ and $d_4 = 180$.

Repeating a BO experiment with a different initialization results in a different regret function and therefore a different convergence time. To take into account different convergence paths (Figure 3.4), I performed 5 BOSS cycles with a different random initialization for each single-fidelity experiment and each multi-fidelity experiment.

3.6.2 Multi-fidelity settings

The ICM model introduces additional hyperparameter that need to be considered. I use the same recommendations from reference [39] and set the hyperparameter prior distribution to

$$p(w) = \mathcal{N}\left(0.9\sqrt{\frac{2}{\beta}}, \frac{1}{2\sqrt{\beta}}\right)$$

where w is used to parameterize the hyperparameters in the matrix $\mathbf{B}$ from equation (2.21). The parameter β in $p(w)$ is chosen as $\beta = 2/\hat{A}^2$, where $\hat{A}$ is again the expected amplitude of the target fidelity.

For both transfer learning and multi-task learning, I used HF as well as UHF as the target fidelity level. When HF was the target fidelity, I used LF as support fidelity simulator. For UHF as target fidelity, I tested both LF and HF as support fidelity. Since there was a large gap between UHF and the other fidelities and a relatively small gap between LF and HF, this allows me to test the impact of the simulator calculation times on the performance of the multi-fidelity approaches.

An objective of the transfer learning experiments was to test what influence the number of support fidelity samples has on the convergence time. In the 2D baseline calculations, all experiments converged within $6-26$ BO iterations, while for 4D all experiments converged within $53-148$ iterations. For the number of support fidelity samples in the transfer learning experiments, I choose twice the number of samples that were required to converge the baseline experiments. Within this range, the global minimum was guaranteed to be found for all baseline calculations. This translated to 50 initialization points for 2D and 200 initialization points for 4D transfer learning experiments. In addition, to check the influence of the number of initialization points, I tested 100 initialization points for the 4D experiments.

For multi-task calculations, I designed 7 different approaches (Section 2.2.3). Some of these approaches were highly experimental, such as approach A2. With the large computational cost of the UHF simulator, testing all the in section 2.2.3 discussed multi-fidelity acquisition strategies becomes unfeasible. To get around this limitation and still have the possibility to test all strategies, I introduce so-called alanine simulator

models. These models are used for prescreening experiments. I conducted experiments to help us decide which approaches to keep and which approaches to reject, meaning that they are not further investigated in further work. The accumulated simulator data from the lengthy baseline experiments can be used to train regression models that mimic the true potential energy surfaces. These models correspond to the alanine simulator models.

The prescreening experiments are carried out only for the 2D system, since this should already make it possible to reject the approaches that show little or no improvement over single-fidelity runs. After the prescreening, the real simulators are applied to the 2D and 4D optimization again.

As the alanine simulator models are GP models, the evaluation time of the alanine simulator is negligible compared to the evaluation time of the real simulators. This means that when we use these simulator models for MFBO experiments, we only have to take into account the computation time required for optimizing the acquisition function and posterior model at each iteration for the computation time of a BOSS experiment. The actual simulation time from the simulators is then added manually when the resulting data is post-processed. By substituting the alanine simulator models with the real simulators, the timing becomes feasible and consequently statistical tests can be carried out for all seven multi-fidelity approaches. I used the results of these tests to decide which approaches should then be tested with the real simulators.

For the 2D alanine simulator model tests, I repeated 10 multi-fidelity BOSS cycles, each starting with 2 initial random points for each fidelity level and continuing with 100 MFBO iterations. I tested the ELCB, MES and the MUMBO acquisition functions. Note that ELCB and MES can be used in the single-fidelity as well as in the multi-fidelity experiments, therefore I tested these AFs with all seven multi-fidelity approaches. MUMBO however, is a pure inseparable multi-fidelity AF, therefore it is only tested with approach A6.

Multi-fidelity approaches use the simulator cost as a parameter in the decision-making for the next sample location. The cost parameters were chosen according to rounded values of the simulation durations obtained in section 4.1. With these values, I expect the multi-fidelity runs with the alanine simulator model to be good approximations of the convergence behavior of the real simulators. I used the same cost parameters for the subsequent multi-fidelity experiments with the real simulators.

For the 2D transfer learning experiments, I set the maximum number of target fidelity BO iterations to 20, for 4D experiments to 80. In multi-task learning, the maximum number of BO iterations has to be increased, as this number includes samples from the target fidelity as well as samples from the support fidelity. For the prescreening

experiments, where I used the alanine simulator model, I set the maximum number of BO iterations to 100, as most of the single-fidelity runs have converged at around 20 iterations. I considered the multi-fidelity approaches that have not found the global minimum after the maximum of BO iterations is reached as not converged (bad accuracy). For the 2D multi-fidelity experiments using the real simulators, I set the maximum to 150 BO iterations, for the 4D experiments, I set a maximum of 300 BO iterations. In 4D optimization, most of the single-fidelity experiments converged after around 120 BO iterations.

4 Computational experiments and results

This chapter starts in section 4.1 with a correlation study between the different fidelity levels. Next, I will discuss the established single-fidelity baseline results on the alanine system in section 4.2, which are used to benchmark the multi-fidelity approaches. Section 4.3 and section 4.4 describe the production experiments and illustrate the convergence statistics of the transfer learning and multi-task learning approaches. The approaches are compared and discussed in section 4.5, which also concludes this chapter with recommendations for the use of MFBO.

4.1 Correlation between fidelity levels

Before conducting the BO experiments, I compared the LF, HF and UHF atomistic simulators (Section 2.3) with each other. Properties that are of interest to us concern the simulation duration as well as the correlation between the simulators. This allows me to later formulate recommendations which simulator combinations (characterized by simulator costs and correlation) are best suited for the greatest computational savings. To determine these quantities, I performed calculations with all three simulators for a number of 100 different samples for the 2D search space and 200 different samples for the 4D search space. These samples were chosen according to the Sobol sequence, which is a sequence following a quasi-random number generator [57]. Since the calculation times of the simulators vary depending on the molecule configuration, I measured the average times as well as the standard deviations of the 100 samples (Table 4.1) for all three simulators.

There is a difference of three to four orders of magnitude (Table 4.1) in the computation time between UHF and the remaining two fidelities. While our force field (LF) and density functional theory (HF) simulations can be performed in a matter of seconds, quantum chemistry (UHF) is much more expensive and has an average evaluation time of about 3.5 h. This huge gap between the simulator times makes the combination of these fidelities particularly interesting for the application of MFBO. Given that UHF is

Table 4.1: Timing statistics for the simulators. Average times and standard deviations are calculated for 100 samples in the 2D search space.

Fidelity	$\bar{t} \pm \sigma_t$ [min]	Relative time to UHF
LF	0.03 ± 0.05	0.02%
HF	0.44 ± 0.04	0.22%
UHF	197.14 ± 82.4	100%

so much more expensive than the other fidelity levels, we want to minimize the need for UHF samples while maintaining the accuracy of the UHF simulator.

Table 4.2 provides an overview of the energy value statistics from the list of Sobol points, both for the 2D and the 4D alanine system. All three simulators were evaluated according to the Sobol sequence, meaning that they were sampled at the same sample locations. The energy values calculated with the simulators can be used to determine the Pearson correlation coefficients, a measure of the linear correlation between two data sets. For the 4D alanine system, the obtained data are observations from a UHF baseline run (see section 4.2) instead of the Sobol sequence, 4D LF and HF were sampled at the same sample locations as the UHF run. This enables the calculation of correlation coefficients also in 4D but creates a bias towards low energy values in the statistics, since baseline runs sample more often in low energy regions than the runs using the Sobol sequence.

Table 4.2: 2D (4D) observed value statistics for $N = 100$ ($N = 200$) samples. 2D data are observations according to the Sobol sequence. 4D data are observations from a sequence obtained from a UHF baseline experiment.

Dim.	Fidelity	Mean	Std.	Min	Max	Amplitude	$P_{25\%}$	$P_{75\%}$
2D	LF	11.52	5.35	0.27	20.53	10.13	7.71	16.23
2D	HF	14.74	6.55	0.09	23.66	11.78	8.47	20.45
2D	UHF	13.00	6.17	0.08	21.61	10.76	7.09	18.66
4D	LF	10.42	7.74	0.22	33.09	16.44	4.24	16.50
4D	HF	10.84	8.16	1.14	29.85	14.35	4.15	16.65
4D	UHF	9.47	7.79	0.00	27.78	13.89	3.10	15.59

Table 4.3: Pearson correlation and covariance matrix for the simulators. Statistics for 2D simulator are evaluated for 100 points according to the Sobol sequence, while 200 points are used for the 4D simulator. UHF and HF share the highest correlation.

2D and 4D Correlation matrix

		LF	HF	UHF
2D	LF	1	0.945	0.948
	HF	0.945	1	0.996
	UHF	0.948	0.996	1
4D	LF	1	0.963	0.968
	HF	0.963	1	0.998
	UHF	0.968	0.998	1

2D and 4D Covariance matrix

		LF	HF	UHF
2D	LF	28.862	33.383	31.584
	HF	33.383	43.281	40.615
	UHF	31.584	40.615	38.444
4D	LF	60.226	61.102	58.645
	HF	61.102	66.884	63.739
	UHF	58.645	63.739	60.934

The correlation coefficients, as well as the covariances between the simulators, are listed in table 4.3. Scatter plots, as shown in figure 4.1, also indicate a strong linear correlation between the data sets.

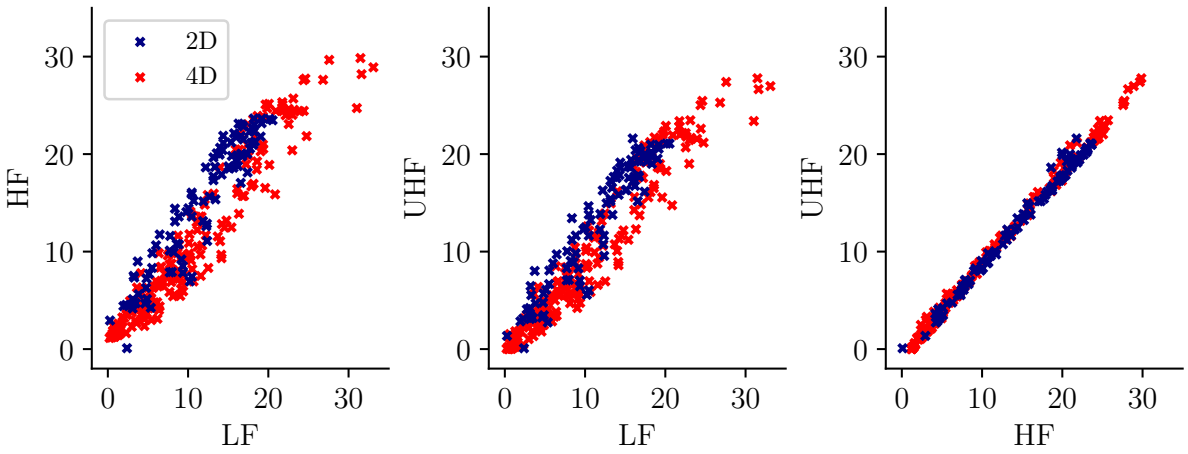


Figure 4.1: Samples from the Sobol sequence. As can be seen, there is a strong linear correlation between UHF and HF, and a good linear correlation between LF and the other two fidelity levels.

4.2 Single-fidelity baseline experiments

The next step was to establish baseline experiments for each fidelity level with single-fidelity experiments against which the multi-fidelity approaches are tested. Baseline

calculations in this context are BO runs where only a single fidelity is used. I performed 5 BOSS cycles, as described in section 3.6, the average convergence time and standard deviation of these runs are then used for the comparison with multi-fidelity experiments. Each of the five cycles was started with two randomly selected samples. I used the ELCB AF for the baseline experiments, since ELCB provides a good trade-off between exploration and exploitation.

For the 2D optimization problem, the converged potential energy landscapes can be compared visually (Figure 4.2). In figure 4.2, it can be seen that the global minimum for

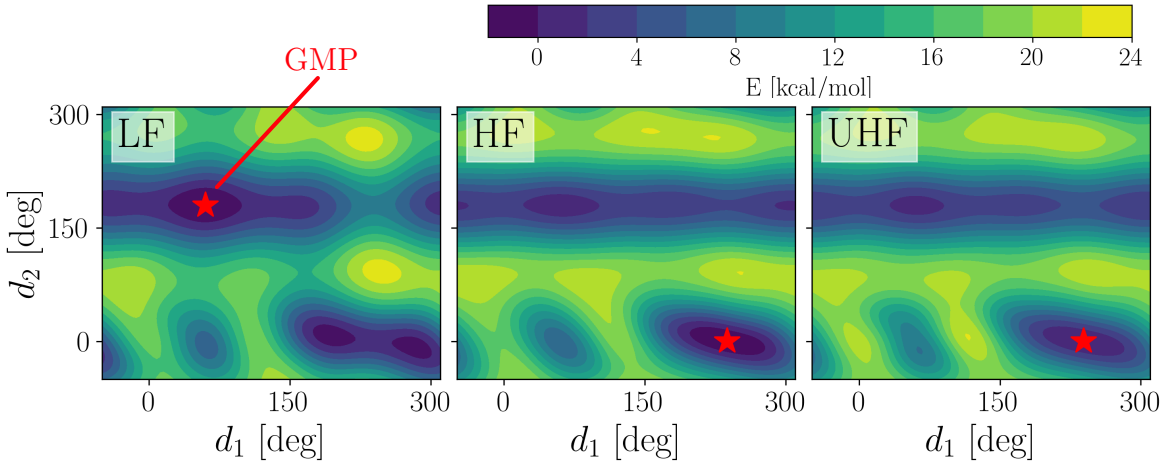


Figure 4.2: Potential energy surfaces of the converged 2D models.

LF is in a different location than for HF and UHF, which makes alanine an interesting test system for the introduced multi-fidelity approaches. All three surfaces are very similar, as can be seen from the figure, which indicates that useful information about the UHF search space can be obtained from the LF and HF simulators.

Table 4.4 contains estimations for the global minima of all three fidelity levels, for the 2D and 4D experiments.

Note that in 2D, the global minimum of HF and UHF are in the same search space location, while the LF global minimum is at another location. Interestingly, in 4D, the roles are reversed: The global minimum of LF is significantly closer to the UHF global minimum than the HF minimum is to UHF. This also allows us to test whether the difference in the location of the global minimum plays an important role in the information exchange of the multi-fidelity approaches.

Table 4.5 contains summary statistics of the CPU convergence times in hours. These

Table 4.4: Estimated global minima. The energy values E are simulator evaluations at the global minima estimations $\mathbf{d}_g$. Dihedral values d_i in parentheses were kept fixed during the optimization. The minima values E are used to set the convergence bounds for BO runs.

Dim.	**Fidelity**	d_1	d_2	d_3	d_4	E [kcal/mol]
2D	LF	59.87	180.07	(60)	(180)	17.482
2D	HF	237.17	0.72	(60)	(180)	-203012.374
2D	UHF	238.95	0.98	(60)	(180)	-202861.338
4D	LF	56.18	179.71	204.74	62.14	15.790
4D	HF	235.91	0.15	61.51	54.94	-203012.440
4D	UHF	56.72	178.79	237.40	60.70	-202861.657

Table 4.5: Summary table of the convergence times in hours for the baseline experiments. The experiments are considered as converged once the global minimum prediction is within an acceptance threshold of 0.23 kcal/mol from the true global minimum.

Dim.	**Fidelity**	**Mean**	**Std.**	**Min**	**Median**	**Max**
2D	LF	0.02	0.01	0.01	0.02	0.02
2D	HF	0.18	0.02	0.15	0.18	0.2
2D	UHF	74.60	9.24	66.17	71.29	88.61
4D	LF	0.17	0.04	0.13	0.15	0.21
4D	HF	1.25	0.21	0.92	1.29	1.50
4D	UHF	214.90	60.28	178.43	180.56	318.98

Table 4.6: Summary table of the BO iterations to reach convergence for the baseline experiments. The experiments are considered as converged once the global minimum prediction is within 0.23 kcal/mol of the true global minimum.

Dim.	Fidelity	Mean	Std.	Min	Median	Max
2D	LF	18.8	7.53	6	23	24
2D	HF	19.8	1.92	17	20	22
2D	UHF	21.6	2.70	19	21	26
4D	LF	60.0	6.12	53	62	66
4D	HF	124.4	19.00	95	125	148
4D	UHF	68.4	18.68	56	59	101

statistics are later used to quantify the speed-up, if any, provided by the different multi-fidelity approaches. In 2D, the convergence times range from about 1 minute (LF), over 10 minutes (HF) up to more than 3 days (UHF). For the 4D systems, times range from about 10 minutes (LF), over 75 minutes (HF) up to almost 9 days (UHF).

The summary statistics for the number of BO iterations required to achieve convergence are shown in table 4.6. These statistics are of interest for choosing a suitable number of transfer learning initialization samples. I used about twice the number of BO iterations that were required to reach convergence as the number of transfer learning initialization points, as in section 3.6 explained. For the 2D experiments, the BO calculations converged for all fidelities on average within about 20 iterations, ranging from 6 up to 26 BO iterations. In the case of 4D optimization, the LF and UHF experiments converged on average after around $60 - 70$ iterations, while HF converged on average after about 120 iterations. The convergences ranged from 53 up to 148 iterations.

4.3 Transfer learning

2D transfer learning experiments

As a first test, I performed 2D MFBO experiments, in which I used HF as well as UHF as the target fidelity level. For HF accuracy experiments I used LF samples (LF→HF) for the initialization, for UHF I set up experiments using either LF (LF→UHF) or HF data (HF→UHF) as support fidelity samples.

Figure 4.3 illustrates the convergence results, both measured in BO iterations (a) and c)) and in CPU time (b) and d)). The statistics for the five runs for each fidelity combination are summarized in the form of a box plot, containing the 25%, 50% and 75% quartile.

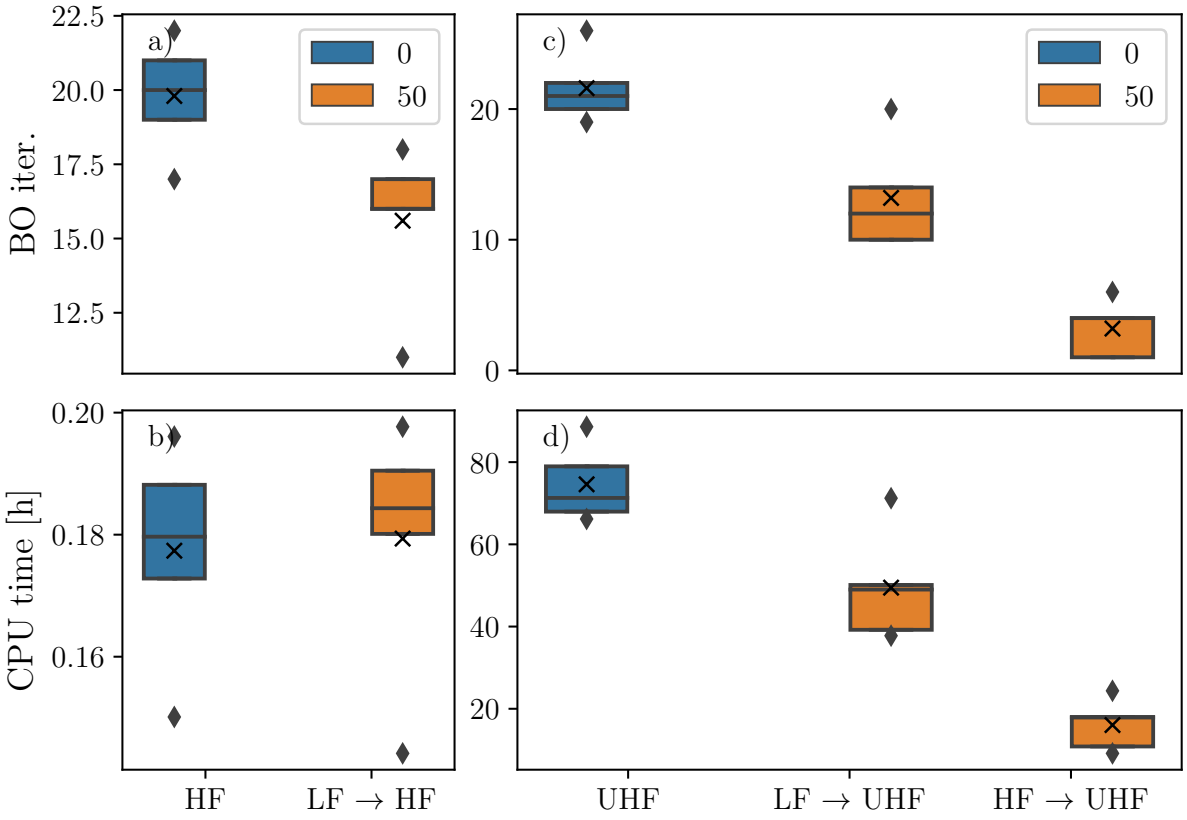


Figure 4.3: 2D Transfer learning results visualized as box plots with percentiles 25 %, 50 % and 75 %. The mean for each experiment is marked by a cross "x", outliers are marked as black diamonds. The number in the legend indicates the number of support fidelity data used at the initialization.

Table 4.7: Summary table of the convergence times in hours for the 2D transfer learning experiments. Bold values indicate the experiments with the fastest convergence times for the respective target fidelity level. TL Init. corresponds to the number of support fidelity samples used for the initialization.

Dim.	Fidelities	TL Init.	Mean	Std.	Min	Median	Max
2D	HF	0	**0.18**	0.02	0.15	**0.18**	0.2
2D	LF → HF	50	**0.18**	0.02	0.14	**0.18**	0.2
2D	UHF	0	74.60	9.24	66.17	71.29	88.61
2D	LF → UHF	50	49.47	13.38	37.78	49.00	71.22
2D	HF → UHF	50	**16.05**	6.15	9.15	**17.87**	24.37

An important result is that all 2D experiments found the global minimum, meaning that all the transfer learning experiments can be considered as maximally accurate.

When HF is the target fidelity (a) and b)), transfer learning improves the efficiency of the number of required BO iterations but increases the CPU times to reach convergence with respect to the baseline. For the scenario where UHF is the target fidelity (c) and d)), the BO iteration and the CPU time show a similar trend of improving the efficiency when transfer learning is applied. We can also note in d) that using the HF simulator for the initialization results in much more savings than using the LF simulator.

Table 4.7 summarizes the statistics for these experiments. The table provides convergence times of the transfer learning and the baseline experiments. For the case of 2D optimization when HF is the target fidelity, baseline and transfer learning have almost similar convergence times. When UHF is the target fidelity, transfer learning always leads to computational savings. For LF as support fidelity, the CPU time decreases on average from about 75 to 50 hours, while for HF as support, the time reduces from about 75 to 16 hours, the lowest observed average convergence time.

4D transfer learning experiments

In the 4D experiments, I used the same fidelity combinations as in 2D. The results are again visualized as a box plot in figure 4.4, where a) and b) show again HF as the target fidelity, while c) and d) contain results where UHF is the target fidelity level. By comparing a) and b), we see the same effect as observed for the 2D experiments: Fewer BO iterations are needed for convergence, but the CPU time shows no savings. In this case, the difference is even more extreme, as the CPU time increases significantly in comparison to the baseline.

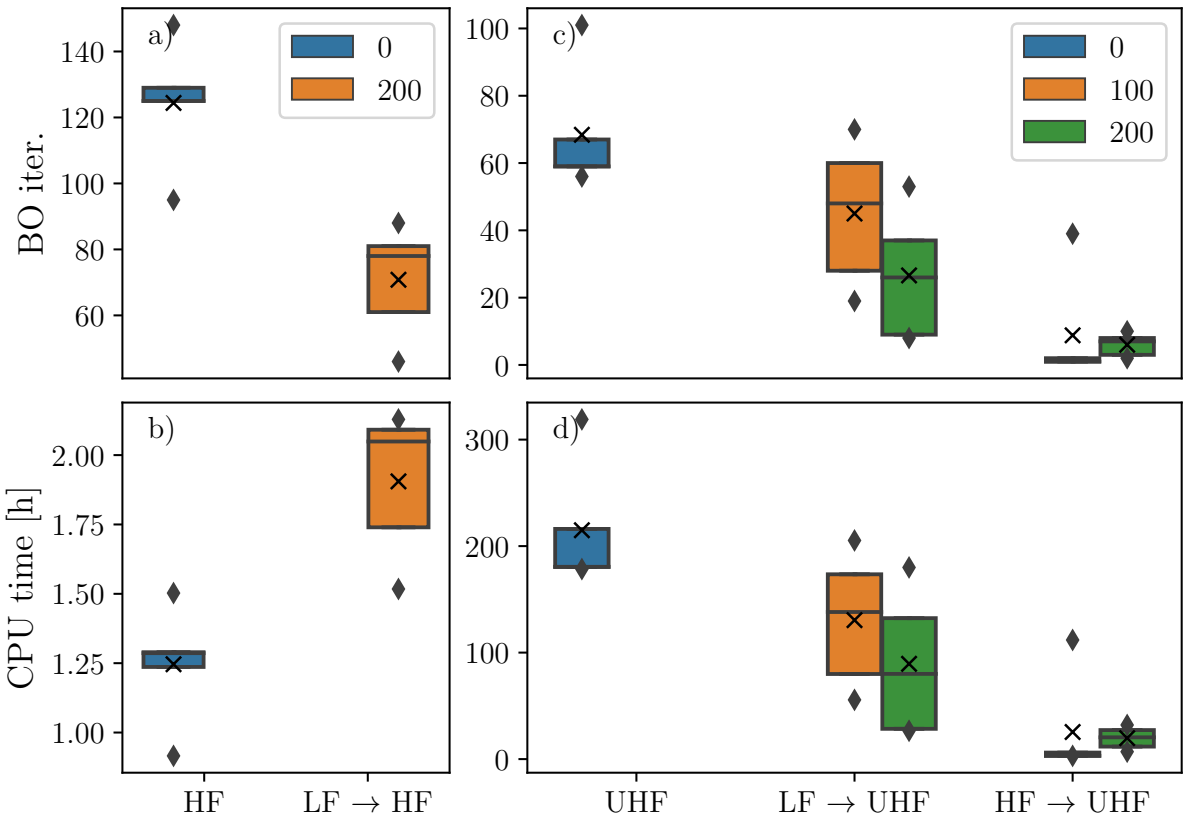


Figure 4.4: 4D Transfer learning results visualized as box plots with percentiles 25 %, 50 % and 75 %. The mean for each experiment is marked by a cross "x", outliers are marked as black diamonds. The number in the legend indicates the number of support fidelity data used at the initialization.

Table 4.8: Summary table of the convergence times in hours for the 4D transfer learning experiments. Bold values indicate the experiments with the fastest convergence times for the respective target fidelity level. TL Init. corresponds to the number of support fidelity samples used for the initialization.

Dim.	Fidelities	TL Init.	Mean	Std.	Min	Median	Max
4D	HF	0	**1.25**	0.21	0.92	**1.29**	1.50
4D	LF → HF	200	1.91	0.27	1.52	2.05	2.13
4D	UHF	0	214.90	60.28	178.43	180.56	318.98
4D	LF → UHF	100	130.49	62.60	55.61	138.12	205.30
4D	LF → UHF	200	89.47	66.70	26.57	80.01	179.96
4D	HF → UHF	100	25.44	48.33	3.02	3.11	111.85
4D	HF → UHF	200	**19.68**	10.43	6.95	**20.45**	32.02

When UHF is the target fidelity level, the trend for the number of BO iterations and the CPU time to reach convergence is again similar, both are improved when transfer learning is applied. Using the HF simulator as support has again, as in 2D, provided more computational savings than using LF as support. For 4D, I tested two different initialization numbers with either 100 or 200 support fidelity samples for the initialization. When LF was the support task, 200 initial points performed significantly better than (on average about 30 BO iterations) using 100 initial points (50 BO iterations). For the scenario where HF was the support, the difference between 100 and 200 initialization points was rather negligible (both on average at around 10 BO iterations).

Table 4.8 contains summary statistics for the convergence times in hours. When HF was the target fidelity, the average convergence time went up from about 75 min to almost 120 min when transfer learning was applied. For UHF as target fidelity, transfer learning requires only a fraction of the computational resources to converge in comparison to the baseline experiments. In the case of 100 initialization points, the time went from 215 hours over about 130 hours (LF support) down to 25 hours (HF support). For 200 points, the timing went from 215 hours to 90 hours (LF support) and down to about 20 hours (HF support), the lowest observed average convergence time.

Table 4.9: Accuracy of the 2D alanine simulator model experiments. Column '#' refers to the approach index. The table lists the number of not converged experiments (out of 10 experiments). Bold values indicate experiments which tend to fail to find the global minimum more often (more often than 20 % of all runs), these strategies can be considered to have a poor accuracy and should therefore be not further used.

AF	#	HF→UHF	LF→UHF	AF	#	HF→UHF	LF→UHF
ELCB	A1	0/10	0/10	MES	A1	1/10	2/10
ELCB	A2	0/10	1/10	MES	A2	0/10	1/10
ELCB	A3	0/10	2/10	MES	A3	0/10	0/10
ELCB	A4	1/10	**8/10**	MES	A4	0/10	**9/10**
ELCB	A5	**3/10**	**9/10**	MES	A5	**3/10**	**10/10**
ELCB	A6	0/10	0/10	MES	A6	0/10	0/10
ELCB	A7	0/10	0/10	MES	A7	0/10	0/10
MUMBO	A6	0/10	0/10				

4.4 Multi-task learning

4.4.1 Prescreening the multi-fidelity approaches

As explained in section 3.6, I introduced alanine simulator models as a prescreening method for the multi-fidelity approaches. I will introduce the short-hand notation 'A1', indicating the multi-fidelity approach 1. To check which of the seven multi-fidelity approaches to keep, I did 2D experiments with 100 BO iterations. All seven multi-fidelity approaches are checked upon if they found the correct global minimum within these 100 iterations. If they did not find the correct global minimum, they are considered as to have poor accuracy. The data shown in table 4.9 indicates that some approaches had often troubles to find the correct global minimum. This was the case for approaches A4 and A5, these approaches are to be considered to have a poor accuracy and should therefore be rejected for further experiments.

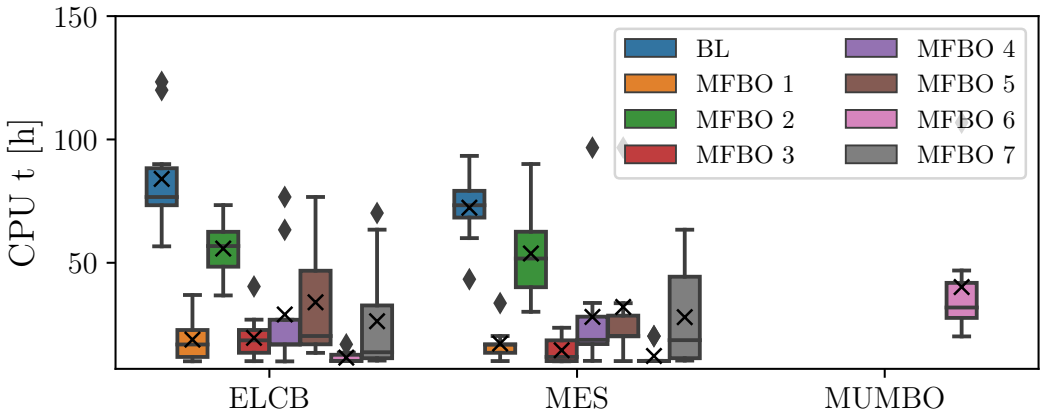


Figure 4.5: 2D multi-fidelity approaches with HF → UHF using the alanine simulator models. CPU time on the y-axis is normalized with respect to the average acquisition time of the UHF simulator (3.5 hours). BL indicates baseline single-fidelity experiments.

Figure 4.5 features the convergence times in CPU hours for the case where HF is used to provide support fidelity data and UHF is used as target fidelity simulator. As the figure illustrates, all multi-fidelity approaches for all AF choices are converging faster than the baseline runs (blue box plot). The approach A2 was a highly experimental approach based on the Knowledge Gradient AF [40] and appears to perform only slightly better than the baseline experiments. Therefore, I decided to not use this approach for the real simulators. A7 has a reasonable average convergence time but exhibits a large variation in the convergence values, same as the A5 for the ELCB AF.

I repeated the experiments with LF as support fidelity, the results are shown in figure 4.6.

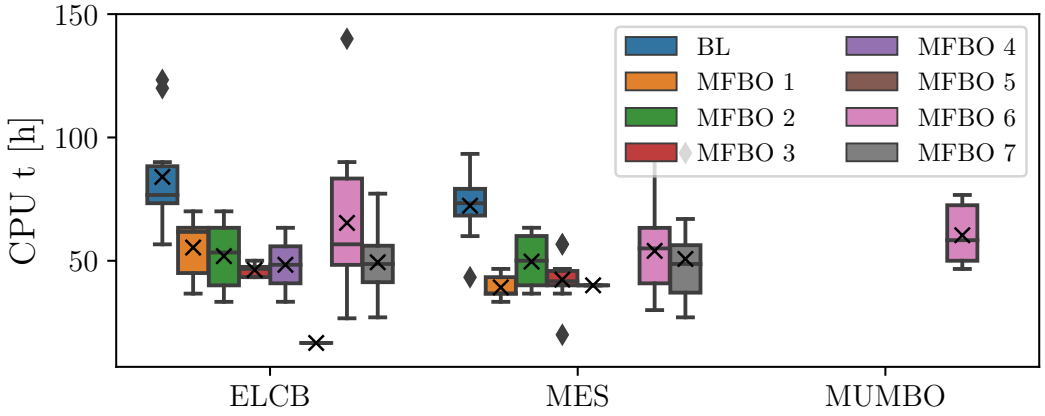


Figure 4.6: 2D multi-fidelity approaches with LF → UHF using the alanine simulator models. CPU time on the y-axis is normalized with respect to the average acquisition time of the UHF simulator (3.5 hours). BL indicates baseline single-fidelity experiments.

In general, the multi-fidelity approaches took longer to converge, which was expected as LF has a weaker linear correlation with UHF than HF has. Here, the A2 strategy, which performed rather poorly before, exhibits a better performance.

In summary, taking all experiments into account, I concluded the following about the acquisition strategies tested:

- A1: Very good efficiency for HF → UHF, good efficiency for LF → UHF.
- A2 : Bad efficiency for HF → UHF.
- A3 : Very good efficiency for both fidelities.
- A4 : Very poor accuracy for LF → UHF.
- A5 : Poor accuracy for HF → UHF, very poor accuracy for LF → UHF.
- A6 : Very good efficiency for HF → UHF, average efficiency for LF → UHF.
- A7 : Average accuracy and large variation of convergence times for both fidelities.

Consequently, I decided to continue the multi-fidelity experiments using only A1, A3 and A6 as those showed to have consistently good convergence results in terms of efficiency and accuracy.

The AF ELCB and MES show only minor differences, while MUMBO tends to perform on average worse than other multi-fidelity strategies. Therefore, I decided to continue the tests on the real simulators using only ELCB as AF.

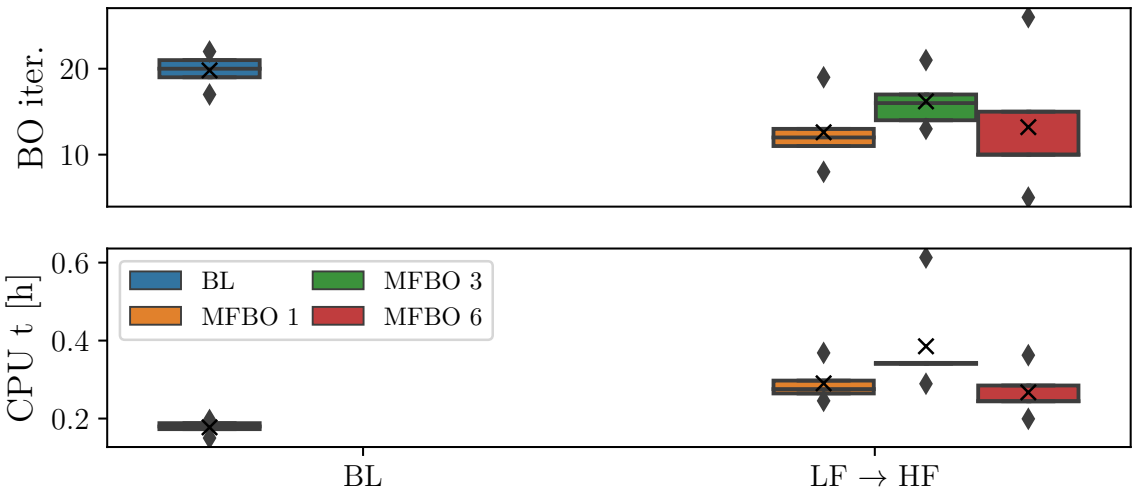


Figure 4.7: 2D multi-fidelity approaches with HF as the target fidelity level. BL indicates the HF baseline experiments. BO iter. corresponds to the number of HF samples until the global minimum prediction has converged.

4.4.2 Tests on real simulators

For the real simulators, I tested again both the optimization over the 2D and the 4D search space. I used the same fidelity combinations as in the previous transfer learning experiments, and also set again the number of repetitions for each experiment to 5, as described in section 4.3. Taking the results from the alanine simulator models into account, I investigated only the approaches A1, A3 and A6 with the ELCB AF.

Figure 4.7 contains the results for the 2D experiments where HF was set as the target fidelity level. As in transfer learning, the number of BO iterations to reach convergence is reduced, while the CPU time for MFBO is larger than for the baseline experiments. There is no significant difference between the multi-fidelity sampling strategies A1, A3 and A6, however the variance of the multi-fidelity approaches is much larger than for the baseline runs.

We obtain a similar result from the 4D experiments (Figure 4.8), when again HF is used as the target fidelity. In comparison to the 2D experiments (Figure 4.7) the difference between the BO iterations and the CPU time to reach convergence is even larger. This indicates that, since the MFBO is again slower than the baseline experiments, the multi-fidelity approaches do not work well for this kind of fidelity combination. Also, the variance is very large again in comparison to the single-fidelity experiment variance.

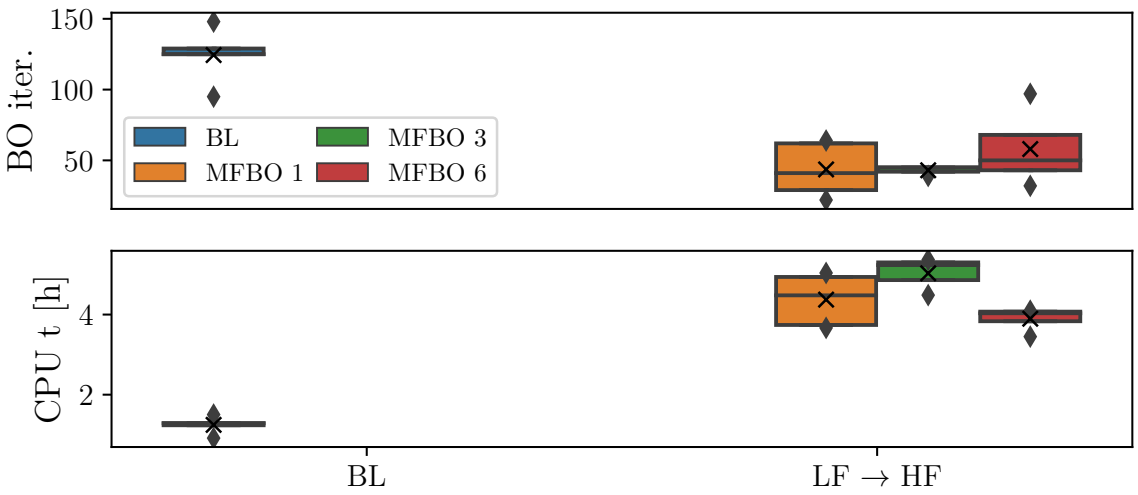


Figure 4.8: 4D multi-fidelity approaches with HF as the target fidelity level. BL indicates the HF baseline experiments. BO iter. corresponds to the number of HF samples until the global minimum prediction has converged.

When UHF is the target fidelity, the convergence results are very different. In 2D (Figure 4.9), it can be seen that the BO iterations as well as the CPU time to reach convergence exhibit a similar trend, as was the case for the transfer learning experiments. The use of LF as support fidelity source reduces the cost again, but the effect is much stronger when HF is used as support fidelity simulator. The correct global minimum was found for all multi-fidelity approaches, except for 1 out of 5 experiments with LF as support for approach A1. All three multi-fidelity approaches are very similar in efficiency, for LF as support, the A1 performs slightly better than A3 and A6. The roles are reversed for HF as support, there, approach A6 performs slightly better than the other two.

The best efficiency was reached for the 4D experiments, when UHF was the target fidelity simulator (Figure 4.10). All fidelity combinations and all multi-fidelity approaches have a much smaller variance than the baseline experiments, which indicates that they are more stable in their convergence behavior. However, 2 out of 5 experiments with LF as support for approach A3 and 3 out of 5 experiments with HF as support for approach A3 experiments have not found the correct global minimum. Interestingly, LF as support fidelity as well as HF as support fidelity both converge quickly compared to the baseline. The summary statistics for the convergence times in 2D and 4D can be found in table A.1 and table A.2.

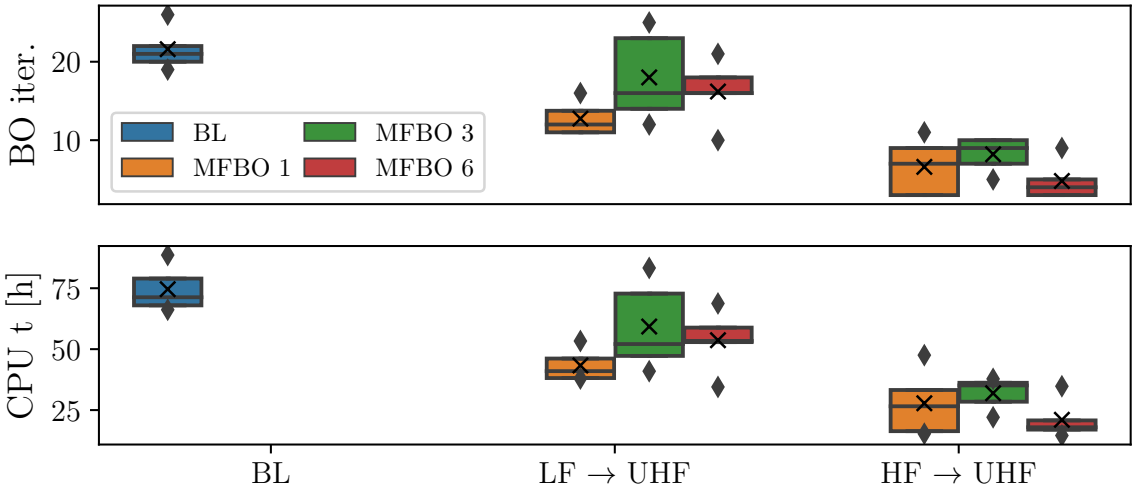


Figure 4.9: 2D multi-fidelity approaches with UHF as the target fidelity level. BL indicates the UHF baseline experiments. BO iter. corresponds to the number of UHF samples until the global minimum prediction has converged.

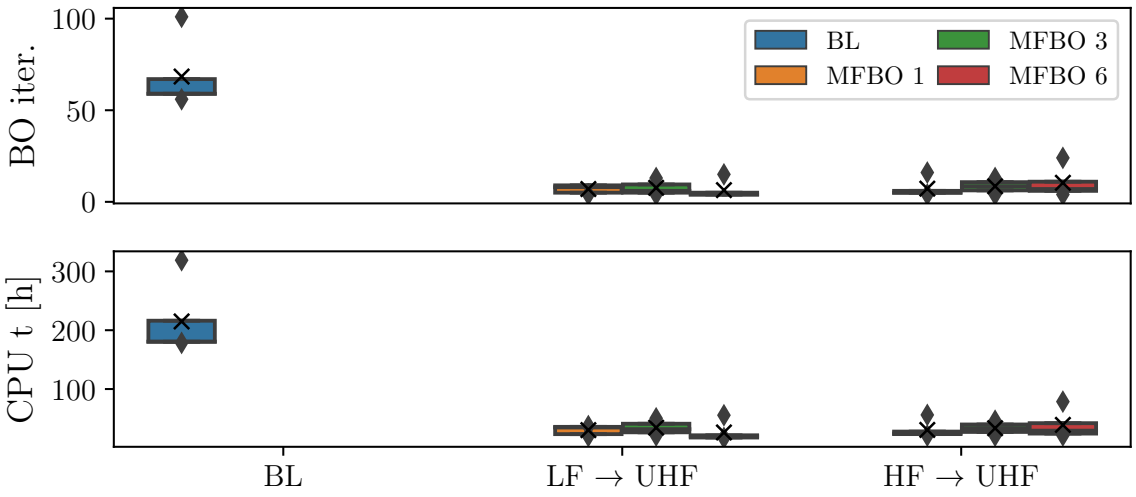


Figure 4.10: 4D multi-fidelity approaches with UHF as the target fidelity level. BL indicates the UHF baseline experiments. BO iter. corresponds to the number of UHF samples until the global minimum prediction has converged.

4.5 Discussion

In the discussion, I will use the obtained results for transfer learning and multi-task learning to answer the formulated research objectives of this thesis (Section 3.5). The first three objectives are answered separately for transfer learning and multi-task learning: These questions related to 1) if a multi-fidelity approach can find the correct global minimum and also how large are the required computational resources with respect to single-fidelity baseline experiments, 2) if using the ICM model for a MFBO approach reduces both, the BO iterations and the CPU time to reach convergence, and also 3) what kind of fidelity combinations are efficient, and if this can be quantified by the correlation and the cost difference between the simulators. For the transfer learning experiments, I will also answer the question of 4) how the convergence statistics change in dependence of the number of support fidelity initialization points. By answering the questions $1-4$, I can conclude the research by directly comparing transfer learning and multi-task learning to see if one of these two approaches is superior over the other in terms of computational savings.

Correlation study between the fidelity levels

To determine the evaluation time of each fidelity and the linear correlation between the simulators, I performed a correlation study before starting the actual single-fidelity and multi-fidelity BO experiments. Table 4.1 provides information for the average timings and standard deviations for a set of 100 simulator calculations for each fidelity level. The timings ranged from about 2 seconds (LF), over half a minute (HF) up to about 3.5 hours (UHF). The standard deviation for the computationally expensive UHF simulator is around 82 minutes, which indicates large fluctuations in the acquisition time depending on the atomistic configuration. The large computation time of 3.5 hours emphasizes the need for a MFBO approach to reduce the number of samples from this simulator as much as possible. Note that the LF and HF simulator are evaluated using a single CPU core, while UHF is evaluated with 40 CPU cores. I did not apply any rescaling to the timings, as the simulation times from LF and HF are already negligible in comparison to UHF within a standard MFBO routine. Also, the timing does not necessarily scale linearly in dependence of the amount of CPU cores used, and in MFBO the samples are taken from different fidelity sources at different steps, so the scaling can't just be applied to the overall time of a MFBO run. Scaling just the sampling times of the UHF and then recalculating the overall times of a BOSS routine in the MFBO scenario would be possible, but would not change the end conclusions.

The ICM models each fidelity level as a linear combination of latent functions (Section 2.2.1), so it can be used to learn fidelity levels that are highly linearly correlated. To determine the linear correlation, I evaluated the three fidelity levels on a fixed set of 100 samples in 2D and 200 samples in 4D. Figure 4.1 illustrates that there is a strong linear correlation between all fidelity combinations, both in 2D and in 4D. The correlation between HF and UHF is particularly strong, which indicates that there is a huge potential to obtain computational savings for that fidelity combination. Table 4.3 provides the Pearson correlation coefficients and the covariances between the simulators. The correlations range from 0.945 (LF and HF in 2D) up to 0.998 (HF and UHF in 4D). The tests to determine the linear correlation coefficients justify the use of the ICM model, which by design is suitable for modelling linearly correlated functions. According to these results, I also expect that the HF simulator as support fidelity will provide better results than the LF simulators. The covariances between the simulators are larger for 4D than for 2D. Also, the amplitude of the energy values obtained in 4D are larger than in 2D (Table 4.2). This can be expected, as the 2D system is a subspace of the 4D configurations space, so it has less possible configurations.

Single-fidelity baseline experiments

The single-fidelity baseline experiments are what would be the standard application of BO on a black-box function. The results from these experiments provide us convergence times, against which the multi-fidelity experiments can be benchmarked, to measure the increase in efficiency of MFBO over BO. Figure 4.2 visualizes the converged energy surfaces in 2D, here it can be seen that the surfaces share very similar structures. In 2D experiments, the convergence times were on average 2 min (LF), to 11 min (HF) up to around 3 days (UHF). For 4D experiments, the times ranged from around 10 min (LF) over 75 min (HF) up to (UHF) almost 9 days (UHF). The fact that it takes even for the rather small molecule alanine up to 9 days to find the global minimum supports the need for a method to speed up the minimum inference at UHF inference. In addition, the UHF convergence times were spread widely: while the fastest convergence time was about 7.5 days, the longest convergence time took up to almost 14 days.

In 2D, the predicted global minimum of LF and UHF are very close to each other (Table 4.4), while in 4D the global minima of LF and UHF are rather close. This helps us to investigate in the multi-fidelity results whether the correlation or the location of the global minimum of the support fidelity is critical to obtain as many computational savings as possible.

Table 4.6 provides the number of BO iterations required to reach convergence. In 2D, the number of BO iterations of all fidelity levels are around 20 and show a small variation

(except for LF, having one BO experiment that converged after only 6 iterations). For 4D optimization, LF and UHF have an average iteration of around 60-70, while HF has an average number of BO iterations of about 120. The variation is small for LF, while it is rather large for HF and UHF. For that reason of large variations in the number of BO iterations, I used twice the number of BO iterations to reach convergence for the initialization of the transfer learning experiments. This should provide enough information about the correct global minimum and also additional information about other local minima of the support fidelity.

Transfer learning experiments

One objective of the transfer learning experiments was to figure out, if, and how fast, the global minimum can be correctly predicted. In all conducted 2D and 4D experiments, the global minimum has been found. Transfer learning decreased the number of required BO iterations to reach convergence in all experiments, for all fidelity combinations and for all investigated numbers of support fidelity samples (50 in 2D, 100 and 200 in 4D). I also wanted to investigate if the ICM model reduces both, the BO iterations and the CPU time to reach convergence. It turned out, that the ICM model only decreases the CPU time as well in the scenario where UHF is the target fidelity. In the case where HF was the target fidelity, the additional computing time caused by the ICM kernel is too large to reduce the computation time. Even though the number of required BO iterations is reduced, the savings get compensated by the additional cost produced by the ICM kernel. Consequently, requiring less BO iterations did not generate any computational savings when HF was the target fidelity. For UHF as target source, the computational cost of the ICM model can be neglected in comparison to the cost of sampling the UHF fidelity level.

In terms of efficiency, how fast the global minimum has been found, HF as a support was better than LF as support. This has been expected, as HF shares a larger correlation with UHF (0.996 in 2D and 0.998 in 4D) than LF does (0.945 in 2D and 0.963 in 4D). In 2D, the convergence time was reduced by up to almost 80 % with HF as support. In 4D, with 200 initialization points, the timing was reduced by more than 90 %.

As discussed in detail in previous work [12, 39], it is not clear how many support fidelity samples should ideally be used to initialize the MTGPR in transfer learning. *Too many* support fidelity samples could “confuse“ or distract[39] the MTGPR model from sampling regions that are of actual interest to infer the target fidelity global minimum. Confusion here means that the sampling strategy after initialization of the model is influenced by support fidelity samples in such a way that MFBO has a worse learning performance than single-fidelity BO. This could occur especially in the case of a weaker correlation

between the fidelities. On the other hand, using *too few* samples can mean that the potential of transferring useful information with the support fidelity samples is not fully exploited. With too few samples, the higher computational complexity of the ICM model could also lead to no savings, although the samples would have useful information.

I tested this using either 100 or 200 initialization points in the 4D optimization task. Figure 4.4 shows that in the case for LF as support fidelity, 200 initialization points provides more savings. Here, the average convergence time was about 130 hours for 100 initialization points and about 90 hours for 200 initialization points. In the case of HF as support fidelity, the difference between the savings reduced drastically. Here, the average time was about 25 hours for 100 initialization points and about 20 hours for 200 initialization points. This suggests that in the case of a very high correlation, such as between HF and UHF, the number of support fidelity points is relatively less important compared to the correlation between the fidelities. For LF as support fidelity, having more initialization points provided better results.

In the case of transfer learning, where the target fidelity is much more expensive than the support fidelities, I would recommend to use about twice the number of samples for the initialization that have been required to find the global minimum in the single-fidelity experiment.

Multi-task learning experiments

Prescreening multi-task experiments

The prescreening experiments (Section 4.4.2) were conducted to reduce the number of approaches and use only those that were effective in terms of accuracy (always finding the global minimum) and efficiency (finding it fast). The results (Figure 4.5 and Figure 4.6) indicate, that all multi-fidelity approaches perform better than the baseline results. However, some of those approaches had a bad accuracy or general relatively worse efficiency. In the following, I will restrict the discussions to the approaches that I rejected from further analysis, as the successful approaches are discussed in more detail when they are used with the real simulators.

The accuracies (Table 4.9) indicated that Approaches 4 and 5 often failed to find the global minimum within the provided computational resources of 100 MFBO iterations. Both of these approaches belong to the separable acquisition rules, where first the fidelity level and then the sample location was chosen (Section 2.2.3). The fidelity level was determined using the global minimum prediction at that iteration, while the sample location was selected by optimizing a standard acquisition function. With this construction, the choice

of fidelity and sample location are unrelated to each other, which might have caused the poor accuracy that we observe with the prescreening experiments.

In terms of efficiency, most approaches seem to perform well. The only exceptions were approaches A2 and A7, both performed worse than the other approaches when HF was the support fidelity (Figure 4.5). A2 approach is based on the idea of the Knowledge Gradient AF: Investigating which next sample location would maximize the difference between the current posterior model and the posterior model where this sample would be added. I investigated this posterior difference of the target fidelity and scaled it by the cost of evaluating a certain fidelity level. The reason is that the posterior model of the target fidelity typically changes less when a sample from a support fidelity level is added. By scaling with the cost, we once again have a quantity for information gain, here 'target fidelity posterior information per cost'. The alanine simulator models indicate that this approach works reasonably well in the efficiency for LF as support, and rather poorly when HF is used. This behavior, why the efficiency relatively decreases when the fidelity combination's correlation is actually increased, can not be well understood. Therefore, I rejected this acquisition rule from further experiments.

Approach A7 had a comparably well efficiency for LF as support fidelity, but the convergence times had a relatively large variation when HF was used as support. Similar as for A2, the efficiency should actually increase when the correlation between the investigated fidelities increases, so I rejected this approach from further analysis as well.

Since the alanine simulator model calculations could be done rather fast, compared to UHF calculations with the real simulator, I also conducted experiments where I used MES and MUMBO as AF. I expected MUMBO to outperform MES and ELCB as it is an AF that is particularly designed for MFBO. However, neither the LF nor the HF as support fidelity resulted in MUMBO outperforming approaches with ELCB or MES. Using MES directly also did not result in any major differences in comparison to ELCB. One possible explanation could be that in the case where the output tasks are strongly linearly correlated, a standard AF in combination with another heuristic for choosing the fidelity level, is sufficient. It would be interesting to see the performance of MUMBO for cases, where there is a much weaker correlation between the fidelities. According to the prescreening experiment results, MES is also a reasonable choice as an AF to obtain a decent performance in MFBO. However, since it did not provide significantly improved results compared to ELCB, I limited myself to ELCB only for further investigations.

As a conclusion, I continued the multi-task experiments using only the approaches A1, A3 and A6 with ELCB, as those have provided the highest efficiency and accuracy for the alanine simulator experiments.

Real simulator multi-task experiments

In the 2D experiments where HF was used as the target fidelity level (Figure 4.7), all three approaches A1, A3 and A6 resulted in savings for the BO iterations but increased the CPU convergence times. This is the same result that we have observed in the previous transfer learning experiment for this fidelity combination. The reason for this is again the usage of the ICM kernel, which comes at an additional computational cost. For HF as primary fidelity, the evaluation time of the HF simulator is not large enough to justify the additional costs of the ICM model. While the baseline converged at around 20 BO iterations, the three approaches converged at around $12-16$ iterations. However, the CPU time doubles relatively to the baseline, increasing from 10 minutes up to around 20 minutes for all three approaches. The same result is obtained when HF is used as target fidelity for 4D optimization (Figure 4.8). While for all three approaches the BO iterations reduces from about 120 to about 60-70 required samples for convergence, the CPU time increases from about 1.5 hours up to $4-5$ hours.

When UHF is used as target fidelity, the BO iterations and CPU times exhibit a very similar trend. For the 2D optimization (Figure 4.9), there is an obvious difference in the savings, depending on which fidelity is used for the support data. When LF is used as the support fidelity, the BO iterations reduce from around 20 iterations to around $12-16$ iterations for the approaches A1, A3 and A6. The CPU times reduce from around 75 hours down to around $45-60$ hours (Table A.1). In the scenario of LF as support, A1 performs slightly better than A3 and much better than A6. One possible explanation why the A6 performs a bit worse for LF as support fidelity could be that this approach does not take the correlation between the fidelities explicitly into account in the sample strategy. This might provide useful information and could improve the efficiency in cases when the fidelities are not strongly correlated.

For the scenario where HF is the support fidelity, the savings are larger again. The number of BO iterations reduces from 20 (baseline) to around 10 iterations, the CPU time decrease from around 75 hours to 20 hours. This implies that in 2D optimization, a strong correlation between the fidelities plays a key role for the savings that can be established. Also, approach A6 outperforms approaches A1 and A3. As HF and UHF share a very strong correlation (0.996), it seems that the correlation between the fidelities does not be taken into account explicitly to obtain a fast convergence.

In the 4D optimization with UHF as target fidelity (Figure 4.10), the BO iterations and CPU time savings follow again a very similar trend. Unlike in the 2D optimization, there is hardly any recognizable difference in the savings between the different support fidelities. This indicates, that for the optimization over a larger search space, the correlation between the fidelities plays a less important role. The average number of required BO

iterations goes from about 60 iterations down to about $10 - 15$ iterations to reach convergence. The CPU timings decrease from around 215 hours to around $20 - 30$ hours. For LF as support, approach A6 works slightly better than A1 and A3, while for HF as support, A1 works slightly better than A3 and A6. Another reason why A6 might have outperformed the other approaches for LF as support is the fact that the global minimum of LF and UHF are closer to each other than the global minima between HF and UHF. Approach A6 does not take the correlation strength into account and might focus more on sampling the same global minimum region, which has been successful in this case.

Comparing multi-fidelity sampling strategies

In this chapter I will directly compare the sampling strategies of the A1, A3 and A6 approaches. Sampling strategies in this context means, how the multi-fidelity AF chooses the sample locations in the search space, as well as which fidelity level is chosen at which BO iteration step.

A visualization of the sampling paths and global minimum predictions makes it possible to interpret the approaches and to identify possible problems that should be worked on to improve the respective multi-fidelity AF. I plotted the sample locations x_i for each dimension i in search space for one out of the five experiments of each multi-fidelity strategy A1, A3 and A6. In addition to the sample locations x_i, I plotted the global minimum prediction $\hat{x}_i$ (current prediction at the respective BO iteration) for each dimension as well.

Figure 4.11 illustrates the sampling strategy for A1. It can be clearly seen that the strategy has phases of exploration (samples that are further away from $\hat{x}_i$) and exploitation (samples that are close to $\hat{x}_i$, e.g. between iteration 18-25 or 40-55). In the beginning of the optimization, the support fidelity simulator is preferred, while occasional samples are taken from the target fidelity simulator at (almost) equal intervals. After about 60 iterations, the sampling has an exploitation phase and then continues to sample LF and UHF in almost alternating steps, before continuing with sampling LF more often again. The long exploitation phase after the 60-th iteration can be interpreted as follows: The MFBO keeps to sample the support fidelity in this region, until the first sample from the target fidelity is taken in the same region. The reason for that is that during the phase where only the support fidelity is sampled, the uncertainty about the target fidelity in this region is not reduced as much as when the target fidelity itself is sampled. Therefore, the algorithm continues sampling there, as the information gain per cost of approach A1 is about reducing the *uncertainty of the target fidelity*. However, this can be problematic in high dimensional search spaces, such that the BO algorithm is 'stuck'

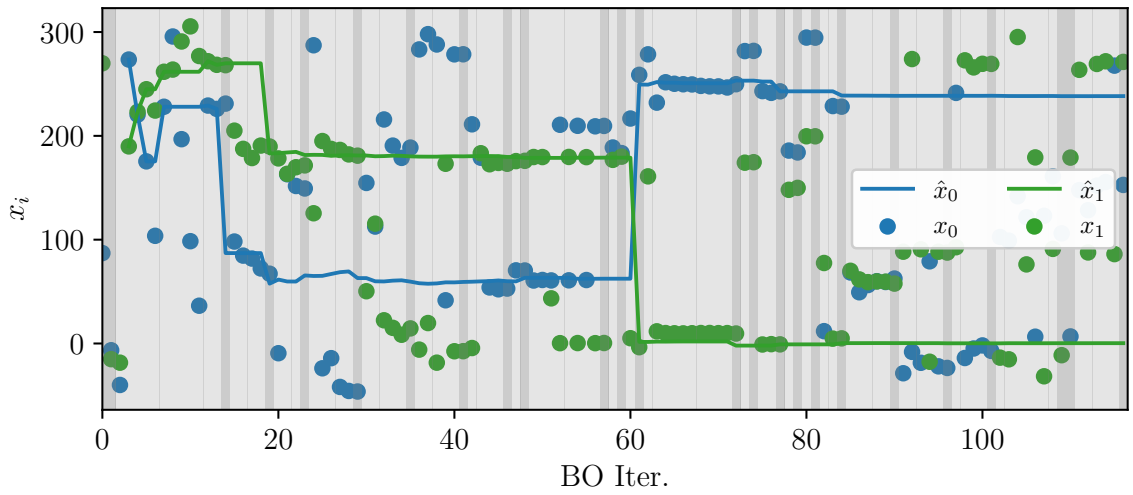


Figure 4.11: Sample locations x_i and global minimum prediction for multi-fidelity strategy A1 for 2D using LF as support fidelity level. Dark backgrounds indicate samples at target fidelity, light backgrounds samples at support fidelities.

for a large amount of iterations. It could follow that in some scenarios, the budget of BO iterations is exhausted, although the computational budget is not fully utilized by sampling only the lower fidelity in such a long exploitation phase. To avoid this, a pure exploration trigger could be used: As soon as the uncertainty of the support fidelity is below a threshold, the exploitation phase could be interrupted and pure exploration activated by evaluating the exploration AF (section 2.1.2).

In strategy 3 (Figure 4.12), similar exploitation and exploration phases can be identified. The support fidelity level is also sampled more often than the target fidelity level, although the target fidelity level is often sampled several times in succession. In this strategy, exploration phases are more prevalent compared to strategy 1.

A bigger difference in the sampling policy can be seen with the strategy 6 (Figure 4.13). Here, only the support fidelity source is sampled for about the first 35 BO iterations. Following that, the strategy continues with only sampling the target fidelity level. This is also what happens in the transfer learning experiments: First, a single-fidelity BO experiment is carried out for the less accurate fidelity level. After this run has converged, the data is used to initialize the ICM model and the MFBO algorithm continues to only sample the target fidelity simulator. In contrast to the transfer learning experiment, here in strategy 6 the number of support fidelity simulator samples does not have to be chosen a priori. It can also be seen that this approach requires fewer overall BO iterations to reach convergence (including the lower fidelity iterations), compared to the previous

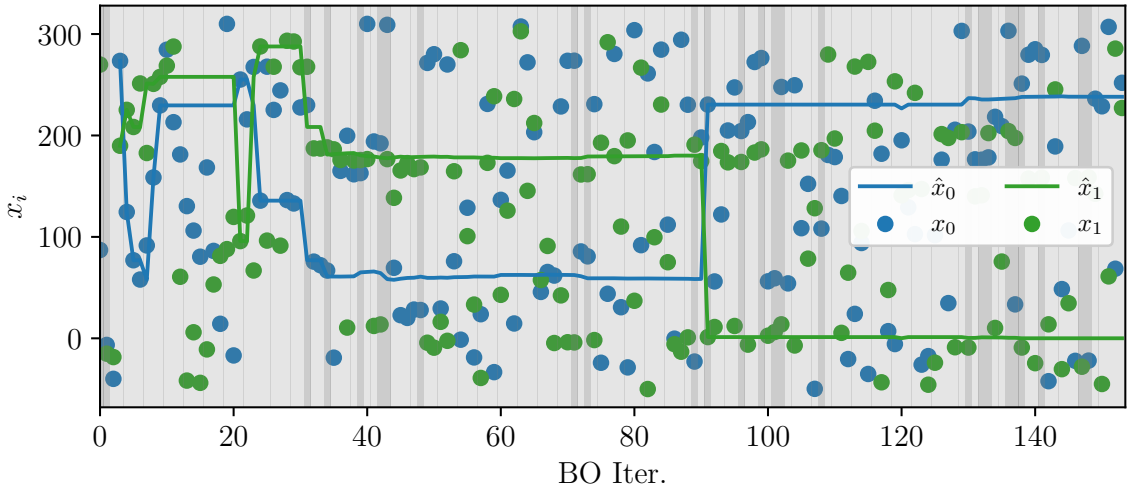


Figure 4.12: Sample locations x_i and global minimum prediction for multi-fidelity strategy A3 for 2D using LF as support fidelity level. Dark backgrounds indicate samples at target fidelity, light backgrounds samples at support fidelities.

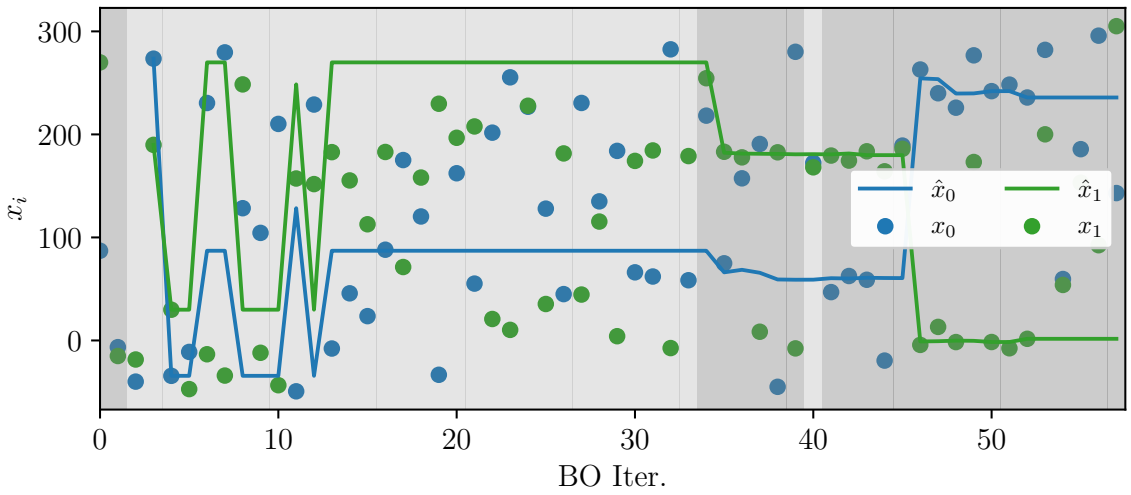


Figure 4.13: Sample locations x_i and global minimum prediction for multi-fidelity strategy A6 for 2D using LF as support fidelity level. Dark backgrounds indicate samples at target fidelity, light backgrounds samples at support fidelities.

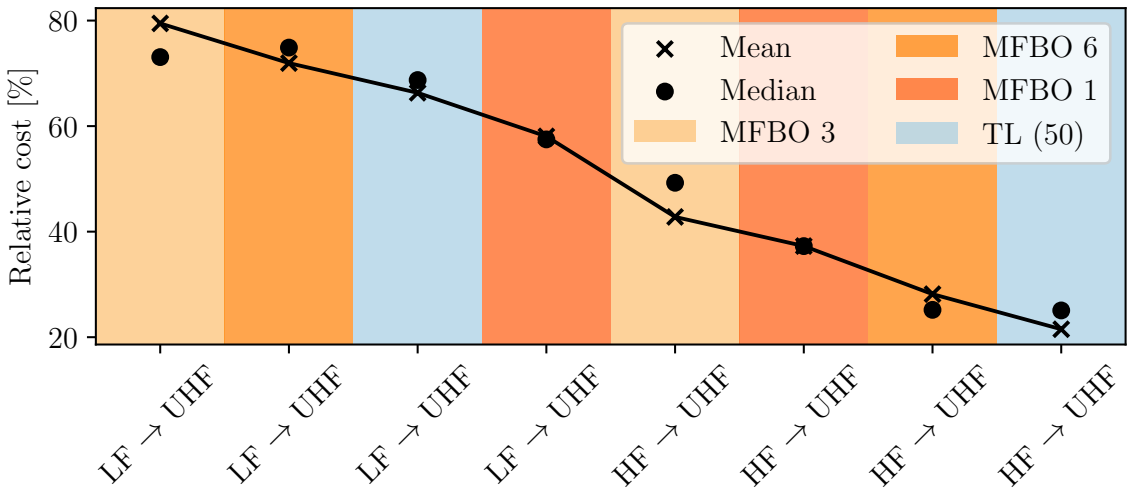


Figure 4.14: Average cost to reach convergence relative to baseline. Experiments for different 2D multi-fidelity with UHF as target fidelity. The x-axis shows the tested fidelity combinations. The blue color indicates transfer learning experiments, red and orange colors indicate different multi-fidelity approaches.

strategies. This can be useful for systems with high-dimensional search spaces, for which the ICM kernel matrix would become very expensive to calculate.

Computational savings comparison of multi-fidelity experiments

Now that I discussed the results of the transfer learning and multi-fidelity learning approaches, I will compare the approaches directly to make recommendations for obtaining the highest possible computational savings. Each of the previous experiments provided statistics for the convergence times. To directly compare the approaches, I ordered all the different experiments by their mean convergence times in descending order. This enables us to identify if there is a trend that indicates that certain approaches work better than others.

In figure 4.14, the results for the 2D multi-fidelity experiments are presented. A clear trend indicates that for 2D experiments, HF as support fidelity source enables much more savings than LF does. Using LF as less accurate source, we can recommend using the multi-fidelity strategy A1, as this results in the greatest CPU time savings. For HF as support fidelity source, transfer learning with 50 initial points provides the best savings from all the conducted experiments. Alternatively, if a user wants to avoid choosing the

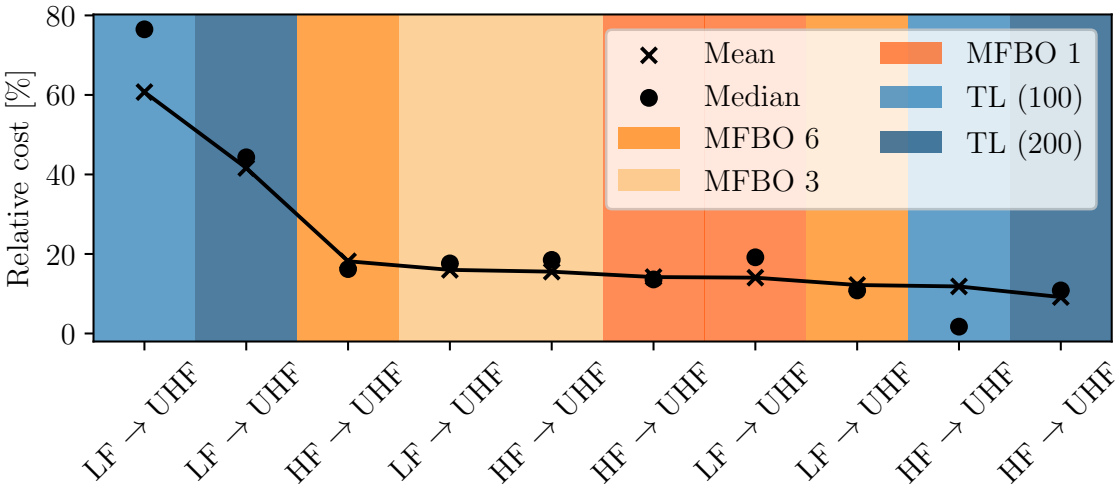


Figure 4.15: Average cost relative to baseline for different 4D multi-fidelity experiments with UHF as target fidelity. The x-axis shows the tested fidelity combinations. Blue colors indicate transfer learning experiments, red and orange colors indicate different multi-fidelity approaches.

number of samples used to initialize transfer learning experiments, strategy A6 can be used, as it results in a comparable cost reduction.

In 4D experiments (Figure 4.15), the situation looks different. Here it is not as clear as in 2D whether LF or HF is more useful for the cost reduction. If a multi-fidelity BOSS user wants to apply a LF simulator as a support fidelity source, the most savings can be expected by using either A6 or A1. For the case where one wants to use HF as support fidelity source, transfer learning with either 200 or 100 initialization points is expected to reduce the cost the most. When, once again, the user wants to avoid setting the amount of initialization points, A6 is recommended.

All the average and median CPU times required to reach convergence are listed in table A.3. In addition to the results in figure 4.14 and figure 4.15, the table also contains the results where HF is used as the target fidelity level. For the scenario where HF is the target fidelity level, single-fidelity experiments should be preferred, as CPU times for the multi-fidelity experiments exceeded those of the baseline experiments.

The best multi-fidelity strategy A6 can be recommended for systems with very strong correlation. For systems where the correlation is not as strong, it might be better to use A1, as this approach takes also the Pearson correlation between the fidelities automatically into account. For atomic systems larger than alanine, the correlation between the fidelities can be weaker, here I would also recommend to use approach A1.

If the amount of BO iterations to reach convergence is known for the support fidelity simulator, transfer learning can be used by using approximately twice the number of samples for the initialization of the ICM model. I would only recommend to use MFBO in BOSS if

- the acquisition times of the fidelity levels are very different,
- if there is a large correlation to be expected between the fidelities, and
- if the average target fidelity acquisition time is significantly larger than the average time that is required to evaluate the acquisition rule and fit the hyperparameter of the ICM model (differently formulated: if the extra computational cost caused by the ICM model can be neglected in comparison to the simulator evaluation cost).

If possible, a user should always use simulators with a higher correlation as the support fidelity, if the evaluation cost of that support fidelity is comparably small to the target fidelity.

Reproducibility of the results

The raw data (BOSS output files) from the simulators of all conducted experiments can be shared upon request. The cleaned and processed data in addition with a Python package that can be used to reproduce the results in this thesis can also be shared upon request. The repository also contains further information on the structure of the raw and processed data and describes how to recreate the analysis. Parts of the data processing scripts were adapted from reference [39]. The code of the multi-fidelity approaches will also be incorporated in BOSS.

5 Conclusions and Outlook

5.1 Research conclusion

Motivation that lead to this work

The objective of this research was to test if transfer and multi-task learning techniques can be used to accelerate BO when applied for the discovery of designs and structures in computational material science. As with many research questions, half of the solution is provided by formulating the research objectives as well as possible.

An important component of materials science is the study of structure in materials. The structure is crucial for many properties, and to study structure there are many experimental approaches. However, experiments in the laboratory can be slow and expensive. As an alternative, we can simulate atomistic configurations in materials, using quantum mechanics theory. This enables us to determine accurate internal energies and properties at the microscopic and macroscopic level. Taking all possible configurations for an atomistic system and the corresponding energies into account, we obtain the so-called potential energy surface. Structures of our interest need to be stable, and these structures reside in low-energy regions on that surface. Depending on the system, energy calculations and therefore simulating the potential energy surface can be very time-consuming. Since we are mostly interested in low-energy regions on that surface, we don't need to study the whole surface. BO is an established machine learning technique that is suitable for the task of efficiently inferring the global minimum of a black-box function. BO provides an algorithm that sample-efficiently learns low-energy regions, and is therefore a well suited technique for this task.

Energy calculations at the quantum mechanical level are still very expensive to carry out. BO provides us an efficient approach, but determining the global minimum at quantum mechanical accuracy can still be unfeasible for many systems, e.g. larger molecules. To keep calculations for such larger systems tractable, we can use energy calculations from less accurate simulators, to which I refer as support fidelity calculations. The problem I approached in this thesis was to use support calculations in the BO algorithm to accelerate the inference of the lowest energy state of the target fidelity simulator. To do

this, I compared a transfer learning and several cost-aware multi-task learning approaches that I designed and implemented.

I tested if transfer and multi-task learning approaches can be used to accelerate the already efficient BO approach even further, and if so, how these approaches should be incorporated in BO. To apply these methods, I used a multi-task GPR model, the so-called ICM model. The ICM architecture is enabling us to learn multiple machine learning models simultaneously and exchange information between the models depending on their *linear* correlation strength. In this work, the different components in the ICM model correspond to potential energy surfaces for each simulator that I used. I benchmarked the different approaches presented in this thesis on an alanine conformer search problem. The simulator distinguished themselves greatly in their computation time, ranging from $1-2$ seconds up to $3-4$ hours, and exhibited a strong correlation since all simulators combinations had a linear correlation greater than 0.9.

Transfer learning objectives and results

In transfer learning, the sampling of different simulators follows a sequential order. A fixed number of samples acquired from a support fidelity BO experiment is used to initialize the ICM model. If the simulators are correlated with each other, this method can also construct a reasonably accurate model for the target fidelity by using support fidelity data. By continuing to apply BO only to the target fidelity, we can expect to require much fewer samples for the inference of the global minimum, while maintaining a high accuracy for the energy.

However, it was not clear if this approach can enable computational savings at all. The ICM model has a higher computational complexity than a standard GPR model, which increases the evaluation time of the model, especially for many simulator samples. For simulator combinations with a weaker correlation or a smaller difference in their evaluation time, there might be no savings possible by using the ICM model. In some cases, the computational cost could be even increased for finding the global minimum of the target fidelity simulator.

The questions I addressed for transfer learning are 1) if transfer learning can enable computational savings, 2) how much computational saving is possible, 3) which transfer learning configurations enable the most savings and 4) what general recommendations can be made for the use of transfer learning. I found out that transfer learning has the potential of **saving up to 90 % computational cost** over conventional BO when applied to alanine with the presented simulators. In all experiments I conducted, transfer learning always decreased the number of required BO iterations to find the

global minimum. However, when the LF simulator is used in combination with the HF simulator, there were no computational savings with respect to the CPU time. In higher dimensional optimization systems, the CPU time even increased significantly. This is due to the fact that the simulator cost difference between these simulators is not large enough, so that the additional computational cost created by the ICM model can not be compensated.

For the quantum chemistry simulator, I found that computational savings resulted every time when transfer learning has been applied. I tested different amounts of initialization data and found that using a larger number resulted in more computational savings. In addition, the transfer learning experiments always found the correct global minimum.

Based on my research results, I recommend the use of transfer learning if the difference in the evaluation times of the simulators is significantly large and if the target fidelity simulator cost significantly exceeds the additional computational costs caused by the ICM model. When BO is applied to lower dimensional configuration spaces ($d = 2$), it is important that the simulators share a very strong linear correlation to obtain significant (up to 80 %) computational savings. For higher dimensional spaces, I found that the correlation can be weaker and would still enable a large amount of savings, which is also consistent with results from previous work [12]. I would recommend to use twice the amount of samples that were required to find the global minimum of the support fidelity simulator as the number of initialization points for transfer learning.

Multi-task learning objectives and results

In multi-task learning, the idea is again to use data from different simulator sources. The key difference to transfer learning is that both fidelity levels are used throughout the BO algorithm. At each BO iteration, the algorithm has to decide not only which location to sample, but also what simulator to use for the next sample. This is the extension of BO to MFBO. I implemented and tested seven different heuristic rules, to decide the sampling source at each MFBO iteration. Similar to the transfer learning scenario, I used the ICM model. In addition, I implemented and tested new acquisition functions, the Max-value entropy search [36] and MUMBO [17] and applied them in the multi-fidelity framework.

The questions to be answered for multi-task learning are about 1) if multi-task learning can enable savings, 2) if so, how many savings are possible, 3) what configurations or approaches for the acquisition function are more efficient, and 4) what recommendations are reasonable for the successful use of multi-task learning. Like in transfer learning, I found sampling strategies that save up to 90 % computational resources while reaching

the same level of accuracy for the global minimum inference. This corresponded to a **reduction from on average 9 days down to about 20 hours** on average. I found that two strategies have a high efficiency and a high accuracy, meaning they result in many computational savings and are consistently finding the correct global minimum. The first approach (approach 1 in this thesis) chooses the next sampling location by a standard acquisition function and decides on the simulator based on which simulator reduces the uncertainty of the target fidelity model the most, scaled by the CPU time it takes to evaluate that simulator. This establishes a trade-off between information of a sample about the target fidelity and the cost of evaluating a simulator at that sample location. The second approach (approach 6) minimizes the acquisition functions for each fidelity level and chooses the sample location and corresponding fidelity, which has the lowest acquisition function value (this approach requires the models to be on the same scale). This approach was also very successful with computational savings, although the simulator correlation is not explicitly taken into account in the acquisition rule. The separable empirical approaches performed better than the theoretically derived multi-task acquisition rule MUMBO. This could have been the case because the fidelities shared a very strong correlation. The acquisition functions ELCB and MES showed a very similar efficiency when applied to a prescreening experiments, where alanine simulators were used.

For configuration spaces with $d = 2$ and highly correlated simulators, I recommend using the approach 6, as this approach has generally enabled more computational savings than the other tested approaches. In the case of $d = 4$, approaches 1 and 6 both provided the most savings. However, approach 6 does not take the Pearson correlation between the simulators into account, which would be important in case the simulators are not as correlated as they were for the tested alanine system. If a user is not sure about the strength of correlation between two simulators, I recommend to use approach 1.

In addition to testing the transfer and multi-task learning approaches individually, another objective was to compare them directly. Whether transfer learning or multi-task learning is more successful depends on the system and the used simulators. Neither method has proven to be consistently dominant in terms of computation savings over the other.

My results demonstrated that for configuration spaces with dimension $d = 4$, multi-task learning worked very well for both support fidelities, while in transfer learning the choice of the support fidelity had a major impact on the savings. Despite this, transfer learning had the most possible savings, both for the 2D and the 4D optimization, however the savings depend very much on the support fidelity choice and the number of initialization points.

5.2 Outlook

The overall conclusion is that multi-fidelity approaches to BO have a very great potential. The results of this work served as a proof of concept that transfer and multi-task learning can enable many computational savings for the application of BO in materials science. Nevertheless, the approaches should be tested more to better understand benefits and potential limitations.

In this work, I used the alanine conformer search problem to benchmark the approaches. Alanine is a rather small molecule, therefore the approaches should also be tested on more complex structures, which have higher dimensional configuration spaces. It would also be beneficial to see the approaches applied to other tasks, e.g. for a surface adsorption task.

The simulators I used had a strong linear correlation, which is a prerequisite for the success of the ICM model. When transfer and multi-task learning are applied to systems with a weaker or not necessarily linear correlation, other multi-task models could be more beneficial. Alternative candidates are to use the more general LMC instead of the ICM model or an auto-regressive model [58], which establishes a hierarchy between different tasks (in this case we could define the hierarchy by the accuracy level of each simulator). In the current ICM approach, the Pearson correlation coefficients are treated like other hyperparameters and get fitted by maximizing the log marignal likelihood. An interesting alternative that might increase the performance is to calculate the coefficients explicitly, this is called the PCM kernel [13].

Further research should also focus more on the entropy acquisition functions, which I did not test with the real simulators. These AFs are theoretically derived instead of heuristically motivated. This does not necessarily mean that these AFs work better, but nevertheless it would still be interesting to see them used in BO. I added the MES and MUMBO AF to BOSS, it would be useful to test these functions also for higher dimensional systems, since I limited myself in this work to apply them to the 2D alanine simulator model. Another promising multi-fidelity acquisition function approach is Gibbon [59], which is implemented in BoTorch [60]. It would be interesting to investigate the performance of this function applied to optimization problems in materials science and to compare this approach directly with MUMBO.

A Convergence statistics

A.1 Multi-fidelity experiments

Table A.1: Convergence times in hours for 2D multi-task learning approaches. The statistics summarize five experiment repetitions of the respective approach. Out of all experiments, only one of the five experiments for LF → UHF A1 has not converged. Bold values indicate the multi-task learning approaches with the most savings with respect to the corresponding fidelity combination.

Fidelities	Approach	Mean	Std.	Min	Median	Max
HF BL	-	0.18	0.02	0.15	0.18	0.20
LF → HF	1	0.29	0.05	0.25	0.28	0.37
LF → HF	3	0.39	0.13	0.29	0.34	0.61
LF → HF	6	0.27	0.06	0.20	0.24	0.36
UHF BL	-	74.60	9.24	66.17	71.29	88.61
LF → UHF	1	**43.31**	7.20	37.94	**40.98**	53.34
LF → UHF	3	59.28	17.98	40.96	52.10	83.831
LF → UHF	6	53.67	12.47	34.50	53.38	68.75
HF → UHF	1	27.80	13.30	15.31	26.56	47.54
HF → UHF	3	31.92	6.57	22.05	35.12	37.81
HF → UHF	6	**21.01**	8.01	14.57	**17.95**	34.78

Table A.2: Convergence times in hours for 4D multi-task learning approaches. The statistics summarize five experiment repetitions of the respective approach. Out of all experiments, 2/5 experiments for LF → UHF A3 and 3/5 for HF → UHF A3 have not converged. Bold values indicate the multi-task learning approaches with the most savings with respect to the corresponding fidelity combination.

Fidelities	Approach	Mean	Std.	Min	Median	Max
HF BL	-	1.25	0.21	0.92	1.29	1.50
LF → HF	1	4.37	0.65	3.67	4.48	5.04
LF → HF	3	5.03	0.48	4.49	5.24	5.37
LF → HF	6	3.90	0.27	3.45	4.03	4.10
UHF BL	-	214.90	60.28	178.43	180.56	318.98
LF → UHF	1	30.17	7.79	20.21	34.60	36.99
LF → UHF	3	34.37	14.75	21.07	31.81	50.23
LF → UHF	6	**26.19**	16.47	16.92	**19.56**	55.49
HF → UHF	1	**30.50**	14.47	20.87	**24.54**	56.02
HF → UHF	3	33.42	17.93	20.64	33.42	46.09
HF → UHF	6	39.05	23.50	21.15	29.25	78.61

A.2 Comparison of transfer learning and multi-task learning

Table A.3: Summary of the convergence times for all multi-task approaches. The times are normalized with respect to the corresponding baseline convergence times. The (*) indicates that the computation time is not exactly 100 % but very close to the baseline duration.

Dim.	Fidelities	TL Init.	Approach	Mean	Median
2D	LF → HF	50	-	*100 %	*100 %
2D	LF → HF	-	1	161.11 %	155.56 %
2D	LF → HF	-	3	216.67 %	188.89 %
2D	LF → HF	-	6	150.00 %	133.33 %
4D	LF → HF	200	-	152.80 %	158.91 %
4D	LF → HF	-	1	349.60 %	347.28 %
4D	LF → HF	-	3	402.40 %	406.20 %
4D	LF → HF	-	6	312.00 %	312.40 %
2D	LF → UHF	50	-	66.31 %	68.73 %
2D	LF → UHF	-	1	58.06 %	57.48 %
2D	LF → UHF	-	3	79.46 %	73.08 %
2D	LF → UHF	-	6	71.94 %	74.88 %
2D	HF → UHF	50	-	21.51 %	25.07 %
2D	HF → UHF	-	1	37.27 %	37.26 %
2D	HF → UHF	-	3	42.79 %	49.26 %
2D	HF → UHF	-	6	28.16 %	25.18 %
4D	LF → UHF	100	-	60.72 %	76.50 %
4D	LF → UHF	200	-	41.63 %	44.31 %
4D	LF → UHF	-	1	14.04 %	19.16 %
4D	LF → UHF	-	3	16.00 %	17.62 %
4D	LF → UHF	-	6	12.19 %	10.83 %
4D	HF → UHF	100	-	11.83 %	1.72 %
4D	HF → UHF	200	-	9.16 %	10.83 %
4D	HF → UHF	-	1	14.19 %	13.59 %
4D	HF → UHF	-	3	15.55 %	18.51 %
4D	HF → UHF	-	6	18.17 %	16.20 %

Glossary

BO

Bayesian optimization: Optimization algorithm used to determine the global optimum of a black-box function for which no functional forms or properties are assumed.

MFBO

Multi-fidelity Bayesian optimization: Extension of the standard Bayesian optimization approach to utilize data from a lower fidelity model to further reduce the optimization cost.

ICM

Intrinsic model of coregionalization: Multi-output Gaussian process regression model in which observed data is represented as a linear combination of a set of latent independent variables that share the same spatial kernel.

LMC

Linear model of coregionalization: Multi-output Gaussian process regression model in which observed data is represented as a linear combination of a set of latent independent variables.

GP

Gaussian process: Stochastic process (a set of random variables indexed by time or space variables) in which every finite collection of random variables is distributed according to a multivariate normal distribution.

GPR

Gaussian process regression: Using a Gaussian process for a regression task, often used to obtain non-parametric machine learning models.

AF

Acquisition function: Heuristic function used in Bayesian optimization to evaluate the usefulness or information of a next sampling candidate to achieve the objective of optimizing a black-box function.

ELCB

Exploratory lower confidence bound: Acquisition function that is constructed using the sum of posterior mean and variance, enables balance between exploitation and exploration.

EI

Expected improvement: Acquisition function that quantifies how much improvement we would expect (improvement means finding a a lower function value than the current lowest observed function value) when sampling at a new location.

ES

Entropy search: Acquisition function that quantifies the information gain about the global minimum location that is expected at a new sampling location.

PES

Predictive entropy search: Acquisition function that quantifies the information gain about the global minimum location that is expected at a new sampling location. Computationally lighter version of the entropy search acquisition function. In this thesis, 'PES' is also used to refer to the potential energy surface, 'PES AF' refers to the acquisition function.

MES

Max-value entropy search: Acquisition function that quantifies the information gain about the global minimum *value* that is expected at a new sampling location.

MUMBO

MUlti-task max-Value Bayesian optimization: Multi-task version of the Max-value entropy search acquisition function that takes the correlation between different tasks into account.

MTGPR

Multi-task Gaussian process regression: Multi-output Gaussian process regression where the outputs are related to each other.

BOSS

Bayesian optimization structure search: A Python library that uses Bayesian optimization for global minimium inference in materials science minimization tasks.

UHF

Ultra high fidelity: Highest level of fidelity used throughout this thesis work, accuracy based on quatum chemistry theory.

HF

High fidelity: Accuracy level based on density functional theory.

LF

Low fidelity: Lowest level of fidelity used throughout this thesis work, accuracy based on force fields theory.

PES

Potential energy surface: The potential energy surface describes the energy of a system such as a molecule in dependence of parameters, often the degrees of freedom. The surface is often used in chemistry and physics to find structures with minimum energy or to determine rates in chemical reactions.

List of Algorithms

Bibliography

1. Garnett, R. *Bayesian Optimization* in preparation (Cambridge University Press, 2022).
2. Kochanski, G., Golovin, D., Karro, J., Solnik, B., Moitra, S. & Sculley, D. *Bayesian Optimization for a Better Dessert* in (Dec. 2017).
3. Letham, B. & Bakshy, E. Bayesian Optimization for Policy Search via Online-Offline Experimentation. *Journal of Machine Learning Research* **20,** 1–30. http://jmlr.org/papers/v20/18-225.html (2019).
4. Ho, Y. C. & Pepyne, D. L. Simple Explanation of the No-Free-Lunch Theorem and Its Implications. *Journal of Optimization Theory and Applications* **115,** 549–570. ISSN: 1573-2878. https://doi.org/10.1023/A:1021251113462 (Dec. 2002).
5. Gubernatis, J. E. & Lookman, T. Machine learning in materials design and discovery: Examples from the present and suggestions for the future. *Phys. Rev. Materials* **2,** 120301. https://link.aps.org/doi/10.1103/PhysRevMaterials.2.120301 (12 Dec. 2018).
6. Batra, R., Song, L. & Ramprasad, R. Emerging materials intelligence ecosystems propelled by machine learning. *Nature Reviews Materials* **6,** 655–678. ISSN: 2058-8437. https://doi.org/10.1038/s41578-020-00255-y (Aug. 2021).
7. Järvi, J., Todorović, M. & Rinke, P. Efficient modeling of organic adsorbates on oxygen-intercalated graphene on Ir(111). English. *Physical Review B (Condensed Matter and Materials Physics)* **105,** 1–10. ISSN: 2469-9950 (May 2022).
8. Järvi, J., Alldritt, B., Krejčí, O., Todorović, M., Liljeroth, P. & Rinke, P. Integrating Bayesian Inference with Scanning Probe Experiments for Robust Identification of Surface Adsorbate Configurations. *Advanced Functional Materials* **31,** 2010853. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/adfm.202010853. https://onlinelibrary.wiley.com/doi/abs/10.1002/adfm.202010853 (2021).
9. Todorović, M., Gutmann, M. U., Corander, J. & Rinke, P. Bayesian inference of atomistic structure in functional materials. *npj Computational Materials* **5,** 35. ISSN: 2057-3960. https://doi.org/10.1038/s41524-019-0175-2 (Mar. 2019).

10. Gopakumar, A. M., Balachandran, P. V., Xue, D., Gubernatis, J. E. & Lookman, T. Multi-objective Optimization for Materials Discovery via Adaptive Design. *Scientific Reports* **8,** 3738. ISSN: 2045-2322. https://doi.org/10.1038/s41598-018-21936-3 (Feb. 2018).

11. Fang, L., Makkonen, E., Todorović, M., Rinke, P. & Chen, X. Efficient Amino Acid Conformer Search with Bayesian Optimization. English. *Journal of Chemical Theory and Computation* **17,** 1955–1966. ISSN: 1549-9618 (Mar. 2021).

12. Pilania, G., Gubernatis, J. & Lookman, T. Multi-fidelity machine learning models for accurate bandgap predictions of solids. *Computational Materials Science* **129,** 156–163. ISSN: 0927-0256. https://www.sciencedirect.com/science/article/pii/S0927025616306188 (2017).

13. Herbol, H. C., Poloczek, M. & Clancy, P. Cost-effective materials discovery: Bayesian optimization across multiple information sources. *Mater. Horiz.* **7,** 2113–2123. http://dx.doi.org/10.1039/D0MH00062K (8 2020).

14. Bonilla, E. V., Chai, K. & Williams, C. *Multi-task Gaussian Process Prediction* in *Advances in Neural Information Processing Systems* (eds Platt, J., Koller, D., Singer, Y. & Roweis, S.) **20** (Curran Associates, Inc., 2007). https://proceedings.neurips.cc/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

15. Remes, U., Todorović, M., Corander, J. & Rinke, P. Multi-task Bayesian optimization in Multi-fidelity Atomistic Structure Search. *in preparation* (2022).

16. Song, J., Chen, Y. & Yue, Y. *A General Framework for Multi-fidelity Bayesian Optimization with Gaussian Processes* in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics* (eds Chaudhuri, K. & Sugiyama, M.) **89** (PMLR, 16–18 Apr 2019), 3158–3167. https://proceedings.mlr.press/v89/song19b.html.

17. Moss, H. B., Leslie, D. S. & Rayson, P. *MUMBO: MUlti-task Max-Value Bayesian Optimization* in *Machine Learning and Knowledge Discovery in Databases* (eds Hutter, F., Kersting, K., Lijffijt, J. & Valera, I.) (Springer International Publishing, Cham, 2021), 447–462.

18. Batra, R., Pilania, G., Uberuaga, B. P. & Ramprasad, R. Multifidelity information fusion with machine learning: A case study of dopant formation energies in Hafnia. en. *ACS Appl. Mater. Interfaces* **11,** 24906–24918 (July 2019).

19. Allaire, D. L., Lam, R. & Willcox, K. E. *Multifidelity Optimization using Statistical Surrogate Modeling for Non-Hierarchical Information Sources* in (2015).

20. Forrester, A. I., Sóbester, A. & Keane, A. J. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **463,** 3251–3269. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rspa.2007.1900. https://royalsocietypublishing.org/doi/abs/10.1098/rspa.2007.1900 (2007).

21. Liu, H., Cai, J. & Ong, Y.-S. Remarks on multi-output Gaussian process regression. *Knowledge-Based Systems* **144,** 102–121. ISSN: 0950-7051. https://www.sciencedirect.com/science/article/pii/S0950705117306123 (2018).

22. Ferná ndez-Godino, M. G., Park, C., Kim, N. H. & Haftka, R. T. Issues in Deciding Whether to Use Multifidelity Surrogates. *AIAA Journal* **57,** 2039–2054. https://doi.org/10.2514%5C%2F1.j057750 (May 2019).

23. Brevault, L., Balesdent, M. & Hebbal, A. Overview of Gaussian process based multi-fidelity techniques with variable relationship between fidelities, application to aerospace systems. *Aerospace Science and Technology* **107.** ISSN: 12709638. arXiv: arXiv:2006.16728v1 (2020).

24. Wang, X., Jin, Y., Schmitt, S. & Olhofer, M. *Recent Advances in Bayesian Optimization* 2022. https://arxiv.org/abs/2206.03301.

25. Hawkins, P. C. D. Conformation Generation: The State of the Art. *Journal of chemical information and modeling* **57 8,** 1747–1756 (2017).

26. Joyce, J. in *Stanford Encyclopedia of Philosophy* (2008).

27. Bass, R. F. *Stochastic Processes* (Cambridge University Press, 2011).

28. Rice, J. *Mathematical Statistics and Data Analysis* ISBN: 9781111793715. https://books.google.fi/books?id=7SI8AAAAQBAJ (Cengage Learning, 2006).

29. Rasmussen, C. E. & Williams, C. K. I. Gaussian Processes for Machine Learning. *Journal of the American Statistical Association* **103** (Mar. 2008).

30. Frazier, P. I. *A Tutorial on Bayesian Optimization* 2018. https://arxiv.org/abs/1807.02811.

31. Brochu, E., Cora, V. M. & de Freitas, N. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *ArXiv* **abs/1012.2599** (2010).

32. Wilson, J. T., Moriconi, R., Hutter, F. & Deisenroth, M. P. *The reparameterization trick for acquisition functions* 2017. https://arxiv.org/abs/1712.00424.

33. Srinivas, N., Krause, A., Kakade, S. M. & Seeger, M. W. *Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design* in *ICML* (2010).

34. Hennig, P. & Schuler, C. J. Entropy Search for Information-Efficient Global Optimization. https://arxiv.org/abs/1112.1217 (2011).

35. Hernández-Lobato, J. M., Hoffman, M. W. & Ghahramani, Z. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. https://arxiv.org/abs/1406.2541 (2014).

36. Wang, Z. & Jegelka, S. Max-value Entropy Search for Efficient Bayesian Optimization. https://arxiv.org/abs/1703.01968 (2017).

37. Alvarez, M. A., Rosasco, L. & Lawrence, N. D. *Kernels for Vector-Valued Functions: a Review* 2011. https://arxiv.org/abs/1106.6251.

38. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. & He, Q. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE* **PP,** 1–34 (July 2020).

39. Sten, N. *Accelerating Bayesian Optimization Structure Search with Transfer Learning* English. Master's thesis (Aalto University. School of Science, 2021), 76+6. http://urn.fi/URN:NBN:fi:aalto-202101311714.

40. Frazier, P., Powell, W. & Dayanik, S. A knowledge-gradient policy for sequential information collection. English (US). *SIAM Journal on Control and Optimization* **47,** 2410–2439. ISSN: 0363-0129 (2008).

41. Atkins, P. *Physical Chemistry* ISBN: 9780716724025. https://books.google.fi/books?id=UXWsQgAACAAJ (W.H. Freeman, 1994).

42. Lewars, E. G. *Computational Chemistry Introduction to the Theory and Applications of Molecular and Quantum Mechanics* 2nd ed. 2011. eng. ISBN: 90-481-3862-0 (Springer Netherlands, Dordrecht, 2011).

43. Salomon-Ferrer, R., Case, D. A. & Walker, R. C. An overview of the Amber biomolecular simulation package. *WIREs Computational Molecular Science* **3,** 198–210. eprint: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wcms.1121. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wcms.1121 (2013).

44. Jones, R. O. Density functional theory: Its origins, rise to prominence, and future. *Rev. Mod. Phys.* **87,** 897–923. https://link.aps.org/doi/10.1103/RevModPhys.87.897 (3 Aug. 2015).

45. Hohenberg, P. & Kohn, W. Inhomogeneous Electron Gas. *Phys. Rev.* **136,** B864–B871. https://link.aps.org/doi/10.1103/PhysRev.136.B864 (3B Nov. 1964).

46. Kohn, W. & Sham, L. J. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review* **140,** 1133–1138 (Nov. 1965).

47. Blum, V., Gehrke, R., Hanke, F., Havu, P., Havu, V., Ren, X., Reuter, K. & Scheffler, M. Ab initio molecular simulations with numeric atom-centered orbitals. *Computer Physics Communications* **180,** 2175–2196. ISSN: 0010-4655. https://www.sciencedirect.com/science/article/pii/S0010465509002033 (2009).

48. Slater, J. C. The Theory of Complex Spectra. *Physical Review* **34,** 1293–1322 (Nov. 1929).

49. Pauli, W. Über den Zusammenhang des Abschlusses der Elektronengruppen im Atom mit der Komplexstruktur der Spektren. *Zeitschrift für Physik* **31,** 765–783. ISSN: 0044-3328. https://doi.org/10.1007/BF02980631 (Feb. 1925).

50. Frisch, M. J., Trucks, G. W., *et al. Gaussian~16 Revision C.01* Gaussian Inc. Wallingford CT. 2016.

51. authors, T. G. *GPyOpt: A Bayesian Optimization framework in Python* http://github.com/SheffieldML/GPyOpt. 2016.

52. Sobester, A., Forrester, A. & Keane, A. *Engineering Design via Surrogate Modelling: A Practical Guide* ISBN: 9780470770795. https://books.google.fi/books?id=ulMHmeMnRCcC (Wiley, 2008).

53. Virtanen, P., Gommers, R., *et al.* SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* **17,** 261–272 (2020).

54. Liu, D. C. & Nocedal, J. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* **45,** 503–528. ISSN: 1436-4646. https://doi.org/10.1007/BF01589116 (Aug. 1989).

55. Paleyes, A., Pullin, M., Mahsereci, M., Lawrence, N. & González, J. *Emulation of physical processes with Emukit* in *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS* (2019).

56. Cao, M., Newton, S. Q., Pranata, J. & Schäfer, L. Ab initio conformational analysis of alanine. *Journal of Molecular Structure: THEOCHEM* **332,** 251–267. ISSN: 0166-1280. https://www.sciencedirect.com/science/article/pii/016612809403943F (1995).

57. Sobol', I. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics* **7,** 86–112. ISSN: 0041-5553. https://www.sciencedirect.com/science/article/pii/0041555367901449 (1967).

58. Requeima, J., Tebbutt, W., Bruinsma, W. & Turner, R. E. *The Gaussian Process Autoregressive Regression Model (GPAR)* 2018. https://arxiv.org/abs/1802.07182.

59. Moss, H. B., Leslie, D. S., Gonzalez, J. & Rayson, P. GIBBON: General-purpose Information-Based Bayesian OptimisatioN. *CoRR* **abs/2102.03324.** arXiv: 2102.03324. https://arxiv.org/abs/2102.03324 (2021).

60. Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G. & Bakshy, E. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. https://arxiv.org/abs/1910.06403 (2019).

Acknowledgement

This thesis was done externally in the computational electronic structure theory (CEST) group at the department of applied physics at Aalto university in Helsinki. I'm very grateful for the excellent supervision by Prof. Dr. Milica Todorović, Dr. Joakim Löfgren, Prof. Dr. Patrick Rinke and Prof. Dr. Jörg Main and for useful tips during my research from Dr. Ulpu Remes. I would like to thank Prof. Dr. Christian Holm for taking over the co-report of this Master thesis.

My biggest thank-you goes to Milica, you have been stuck with me from day one where I joined the CEST group. I always looked forward to our weekly meetings, you have been a fantastic advisor right from the beginning. You guided me very well through this thesis, and I learned a lot from you about how to approach a research project and how important it is to communicate science and results effectively. I learned many things from you that will help me on a personal and on a professional level for a long time to come. I would also like to thank Joakim for the interesting conversations we shared and the valuable advice for my research, for proof-reading my thesis and many insightful tips regarding the development work on BOSS. I'm also thankful for the advices I received from Ulpu. You gave me a lot of very useful tips and references for multi-fidelity Bayesian optimization during our biweekly BOSS developer meetings. I'm also thankful to Patrick. Thanks to you, I was able to do my master's project in the CEST group. I am very grateful to you for this opportunity and for your supervision of my thesis. I am also thankful to Jörg for advising me and enabling me to do this Master thesis project abroad in Helsinki. Without the trust you put in me and without the extra organizational effort you, Patrick and Jörg, put into this project, this master thesis would never have came about. The time that I spent in Finland, where I was part of the CEST group almost the entire time, was very formative for me and amongst the best times of my life (consider that this was during the pandemic!). I would like to thank everyone, including those not mentioned here, who have contributed to this experience.

Lastly, I want to thank my sister Miri and my mum for always picking me up from the airport whenever I was coming home and to my dad and my grandparents for providing me food during that time. And of course for all the other support you provided me during this stay abroad.

Declaration

I affirm,

- that I have written this Master's thesis by myself,
- that I have not used any sources other than those indicated and that I have marked all statements taken verbatim or in spirit from other works as such,
- that the submitted work has not been the subject of another examination.

Helsinki, July 20, 2022 *Manuel Kuchelmeister*